TECHNISCHE
UNIVERSITÄT
DARMSTADT

ACTIVE INTRUSION DETECTION FOR WIRELESS MULTIHOP NETWORKS

Am Fachbereich Informatik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertationsschrift

von

ING. RODRIGO DANIEL DO CARMO

Geboren am 20. Juli 1986 in Comodoro Rivadavia, Argentinien

Erstreferent: Prof. Dr.-Ing. Matthias Hollick
Korreferent: Prof. Salil Kanhere, Ph.D.

Tag der Einreichung: 14. Oktober 2014
Tag der Disputation: 15. Dezember 2014

Darmstadt, 2014
Hochschulkennziffer D17

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmitteln verfasst habe. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch nicht zu Prüfungszwecken gedient.

Darmstadt, den 14. Oktober 2014

Ing. Rodrigo Daniel do Carmo

ABSTRACT

This work focuses on network security and introduces an active-probing technique for intrusion detection in wireless multihop networks. Wireless networks have been the revolution of personal communications of the past decades. Millions of devices with wireless capabilities are sold to end customers every year: smartphones that enable access to the Internet almost everywhere, computers with wireless connections, personal watches, sports shoes, digital cameras, and even lenses with wireless capabilities. Today's communication capabilities are based on the concept of single hop networks. The future and vision of wireless communications is to let radio devices form multihop networks. Wireless multihop networks can be instrumental for several scenarios, such as search and rescue in disaster areas, thus potentially helping to save lives. Detecting attacks that can disrupt the network operation can be considered of utmost importance for many applications.

We found that the field of intrusion detection for wireless multihop networks is generally limited. Purely centralized intrusion detection systems are ill suited because wireless multihop networks miss a clear line of defense due to their distributed nature. The most studied approach to intrusion detection in wireless multihop networks is distributed intrusion detection, which consists of deploying detection sensors in all or part of the nodes of the network. Many different approaches following this distributed detection paradigm have been proposed in the literature, but there is still a lack of practical implementations. Some implemented systems showed that the overhead generated on the nodes is excessively high, which makes the distributed detection principle unsuitable for networks operating on resource-constraint devices. Moreover, passive eavesdropping of the wireless medium makes the detection of certain attacks difficult or impossible. Other proposed intrusion detection/mitigation systems require modifications to the base routing protocols in use, thus braking the compatibility with legacy systems.

In this dissertation we propose a novel approach to intrusion detection that overcomes the limitations of the existing approaches. Instead of deploying intrusion detection systems on the nodes of a wireless multihop networks, we propose deploying nodes into the network only for this purpose. The intrusion detection nodes are trustworthy, as well as less limited in resources (computing resources, energy resources, mobility). In contrast to distributed intrusion detection systems in the literature, the intrusion detection nodes in our scheme do not detect attacks locally but detect them by looking into the nodes of the network from the outside. To this end, we propose we propose employing an active-probing technique in this dissertation. It uses an approach similar to the classical ping or active fingerprinting, i.e., it works by sending testing packets to a host and recording/analyzing its reaction. In addition, if we let the intrusion detection node be mobile, it can move through the network and examine all its nodes. The innovation of our approach is that, in contrast to fingerprinting or classical active techniques, we propose an active technique not to identify or characterize nodes, but to determine if they work according to protocol specifications, as well as to detect malicious activities.

In this dissertation, we propose an active-probing technique for intrusion detection and show its conception, design and evaluation. While remaining general, we test our active-probing technique in an office wireless mesh testbed running the emerging mesh standard IEEE 802.11s. We perform a solid evaluation of the active probing and its parameters. For example, we show how transmitting replications of testing packets can be beneficial under different network conditions and keeps false positive rates below 1.3%. We show how our active-probing technique detects attacks such as selective dropping of packets, black hole and colluding misrelay attacks with detection rates above 90%. We model a temporal-selective Bayes classifier to infer the state of a network node under test which is generally applicable to other systems. For our purpose, it classifies whether a node misbehaves based on the outcome of a set of active probes. We design a recursive probe selection scheme based on the current

posterior of the Bayes classifier and a prediction step. This facilitates reducing the number of active probes while maximizing the insights gained by the set of probes executed. We also show that other passive techniques can complement an active-probing-based intrusion detection system. We further propose a lightweight metric called neighbor variation rate for anomaly detection. We study how the neighborhood varies over time and we represent it with a quantitative measurable value. We create a detection model based on this metric and we apply it for anomaly/intrusion detection. Last but not least, we provide the community with a modular implementation and documentation of our active intrusion detection system for wireless mesh networks as open source software available for free online.

ZUSAMMENFASSUNG

Diese Dissertation beschäftigt sich mit der Angriffserkennung in drahtlosen Multihop-Netzen. Drahtlose Netze haben fraglos eine der wichtigsten Revolutionen persönlicher Kommunikation in den letzten Jahrzehnten ermöglicht. Abermillionen von Geräten mit drahtlosen Kommunikationsfähigkeiten werden jedes Jahr an Endnutzer verkauft: Smartphones, die den Internetzugriff fast überall erlauben, Computer mit drahtloser Konnektivität, Uhren, Sportschuhe, Fotokameras und sogar Brillen mit drahtloser Verbindungsmöglichkeit. Das heute vorherrschende Kommunikationsparadigma für drahtlose Netze basiert auf der direkten Verbindung von Endgeräten mit Funk-Basisstationen (sog. *Single Hop Netze*). Die Zukunft und die Vision für die drahtlose Kommunikation besteht darin, dass mobile Endgeräte zusätzlich zu der Single Hop Kommunikation darüber hinaus in der Lage sind, indirekte Verbindungen über mehrere Zwischenknoten aufzubauen (sog. *Multihop Netze*). Solche drahtlosen Multihop Netze können in verschiedenen Szenerien nützlich sein: Sie können beispielsweise bei der Suche und der Rettung von Menschen in Katastrophengebieten eingesetzt werden, wenn infrastrukturbasierte Kommunikation nicht mehr zur Verfügung steht. In diesem Kontext ist die Sicherheit des Netzes und konkret die Erkennung von Angriffen von hoher Relevanz.

Der heutige Stand der Technik zur Angriffserkennung für drahtlose Multihop Netze weist eine Reihe Einschränkungen auf. Rein zentralisierte Angriffserkennungssysteme sind für drahtlose Multihop Netze nicht geeignet, da diese wegen ihrer verteilten Natur keine klare Verteidigungslinie bieten. Die Angriffserkennungsmethode, die bisher am häufigsten für drahtlose Multihop Netze vorgeschlagen wurde, ist die verteilte Angriffserkennung. Diese basiert auf dem Einsatz von Erkennungssensoren in einzelnen oder in sämtlichen Netzknoten. In der Literatur werden eine Vielzahl verschiedener theoretischer Ansätze vorgeschlagen, die diesem Erkennungsparadigma folgen. Eine zufriedenstellende praktische Implementierung fehlt bisher jedoch. Resultate für die wenigen vorhandenen praktisch umgesetzten Systeme zeigen, dass der Aufwand zu hoch für einen regulären Betrieb ist. Die verteilte Erkennung ist für Netze, die mit ressourceneingeschränkten Geräten operieren, daher nicht geeignet. Des Weiteren macht das bei bisherigen Lösungen überwiegend vorgeschlagene passive Lauschen im drahtlosen Medium die Erkennung einiger Angriffe schwer oder gar unmöglich. Andere vorgeschlagene Angriffserkennungs- oder Angriffsminderungssysteme erfordern Änderungen an den Routing-Protokollen. Dies führt zum Kompatibilitätsverlust mit bestehenden Systemen.

In dieser Dissertation schlagen wir eine innovative Angriffserkennungsmethode vor, die die Einschränkungen aktueller Methoden aufhebt. Anstatt des Einsatzes von Angriffserkennungssystemen auf den einzelnen Knoten im Multihop, nutzt unsere Lösung dedizierte Angriffserkennungsknoten. Diese sind vertrauenswürdig und besitzen hinreichende Ressourcen (Rechenleistung, Energieressourcen, steuerbare Mobilität). Im Gegensatz zu verteilten Angriffserkennungssystemen in der Literatur, erkennen die Angriffserkennungsknoten in unserem Schema die Angriffe nicht ausschließlich lokal, sondern untersuchen die Netzknoten von außen. Für diesen Zweck haben wir eine neuartige Technik auf Basis aktiver Angriffsanalyse entwickelt. Diese Technik nutzt ein Vorgehen, das dem klassischen ‚Ping' oder dem aktiven erstellen von Geräteprofilen (sog. *Fingerprinting*) ähnlich ist. Die aktive Angriffsanalyse sendet sogenannte Testpakete zu dem zu analysierenden Netzknoten, zeichnet dessen Reaktion auf und analysiert diese. Darüber hinaus erlauben wir den Angriffserkennungsknoten sich durch das Netzwerk zu bewegen und so alle Knoten zu analysieren. Die Innovation unseres Ansatzes besteht darin, dass unsere aktive Technik im Gegensatz zum Fingerprint oder anderen klassischen aktiven Techniken die Knoten nicht charakterisiert, sondern analysiert, ob die Knoten entsprechend der Protokoll-Spezifikationen funktionieren bzw. ob von ihnen bösartige Aktivitäten ausgehen.

In dieser Dissertation schlagen wir eine Technik zur aktiven Angriffsanalyse für die Angriffserkennungen in drahtlosen Multihop Netzen vor und beschreiben ihre Konzeption, ihr

Design und ihre Evaluierung. Ohne Beschränkung der Allgemeinheit testen wir unsere aktive Angriffsanalyse in einer drahtlosen vermaschten Testumgebung, die dem neuen Standard für Multihop Netze IEEE 802.11s entspricht. Wir führen eine ausführliche Evaluierung unserer Technik und ihrer Parameter durch.

Ergebnisse unserer Untersuchung sind, dass das wiederholte Versenden von Testpaketen in verschiedenen Netzwerkzuständen von Vorteil ist; mit dieser Technik können Falsch-Positiv-Raten von unter 1.3% erzielt werden. Wir zeigen, wie unsere aktive Angriffsanalyse Angriffe wie beispielsweise fortgeschrittene *„selective dropping of packets"*, *„black hole"* und *„colluding misrelay"* mit Erkennungsraten über 90% erkennt. Wir modellieren einen temporär-selektiven Bayes-Entscheider, um daraus den Zustand des getesteten Netzwerkknotens zu schlussfolgern. Dieser Entscheider ist auch für andere Systeme nutzbar. In unserer Anwendung klassifiziert er aus einer Menge von aktiven Proben, ob ein Knoten bösartig agiert. Wir entwickeln ein rekursives Probeauswahlschema, das auf dem aktuellen Posterior und der Vorhersage basiert. Es ermöglicht die Minimierung der Anzahl von aktiven Proben, während die Erkenntnisse aus der Menge von den ausgeführten Proben maximiert werden können. Wir zeigen zudem, dass andere passive Techniken ein aktives Angriffserkennungssystem ergänzen können. Wir schlagen eine leichtgewichtige Metrik für die Anomalieerkennungen vor, die sogenannte *„neighbor variation rate"*. Wir erforschen, wie sich die Nachbarschaft während der Zeit verändert und drücken diese Veränderungen in einer quantitativ messbaren Metrik aus. Wir entwickeln ein Erkennunsmodell, das auf dieser Metrik basiert, und benutzen es für die Anomalie- und Angriffserkennung. Abschließend stellen wir eine modulare Implementierung unseres Systems zur aktiven Angriffsanalyse als Open-Source-Software zur Verfügung.

ACKNOWLEDGMENTS

My doctoral study has been a long road where I crossed a lot of great people, including colleagues and new friends. *My sincere thanks to everyone who I met in this road. Thank you for the unforgettable moments!*

Directly related to my doctoral studies, I express my deepest gratitude to my adviser. Thanks a lot *Matthias* for your constant supervision, for the long discussions, and for your enormous support and patience. I also express my warm thanks to *Salil* for his support and recommendations. A sincere thanks to the committee members of my defense, professors *Alejandro Buchmann, Ph.D.*, *Dr. rer. nat. Oskar von Stryk* and *Dr.-techn. Stefan Katzenbeisser*.

It was a big pleasure to be part of the DFG Research Training Programm GRK Nr. 1362, *Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments (GKmM)* and to work at the *Center for Advanced Security Research Darmstadt (CASED)*. Furthermore, a warm thanks to all my colleagues of *SeeMoo*!

A special thanks go to my family for their great support, especially to my parents *Daniel* and *Viviana*. A special thanks to you *Kathi* for making this road a bit easier.

Darmstadt, 2014 Rodrigo

CONTENTS

1 INTRODUCTION

> To do successful research, you don't need to know everything, you just need to know of one thing that isn't known.
>
> Arthur Schawlow

1.1 MOTIVATION

Practical solutions to provide network-level security in Wireless Multihop Networks (WMNs) are scarce despite the fact that WMNs are a candidate technology for cost-effective wireless network access [1]. The growing relevance of WMNs is mirrored by community efforts that aim at building organically growing mesh communities [1, 2] as well as research projects that investigate technical limitations of wireless multihop networks [3]. Moreover, the recent standardization of the IEEE 802.11s mesh extension aims at building on the success of more than a billion devices with built-in 802.11 capability [4].

Wireless multihop networks are attracting attention and interest in the industry due to their characteristics. They are easy to deploy, cost-effective, and they feature decentralized wireless connectivity that enables communication when a fixed infrastructure is not possible or damaged. An emerging practical example is the communication in disaster areas using Android smartphones [5]. In a disaster area, for example after an extreme meteorological event, the complete communication system might be destroyed. The people, however, still need to communicate, rescue teams need to know where survivors are, neighbors want to contact their families and friends, or send emergency requests when food and water is scarce. Rescue robots or drones might be deployed and they also need to communicate. In such scenario, a wireless multihop network can be quickly and easily deployed to enable communication. Autonomous nodes with battery or solar energy can be installed and create a reliable multihop network that, at the very end, can save peoples' lives. Apart from catastrophe scenarios, other emerging examples include: sensors for industrial automation and medical monitoring [6], car-to-car communication [7], lighting control [8], and community networks for Internet access [9]. Wireless multihop networks are also emerging for smart home applications and the Internet of Things. A group of industry-leading companies developed Thread in 2014, a new IPv6 networking protocol for low-power smart homes [10].

Yet, in spite of their advantages, WMNs present a potential point of failure: each node should *cooperatively* forward the packets to and from neighboring nodes. Because of the meshed topology and multihop transmissions, no centralized monitoring or management unit exists that allows supervision of the forwarding of packets in a network wide-fashion. Once a packet is transmitted by a node, its neighbors are responsible for forwarding the packet to its destination. This underlying principle of collaboration opens the door to malicious nodes to perform a wide range of attacks.

1.1.1 *The security problem*

Security is a general and very important requirement for any kind of computer network. The different applications of a network feature diverse security requirements, yet it is obvious that applications such as rescue systems, medical monitoring, industrial automation or vehicular communications can only be successful if operational security can be maintained. But achieving operational security in wireless multihop networks is not trivial and it should be tackled in

an integrated fashion. An integral security design can be wide and consist of a diversity of requirements, policies, monitoring systems, prevention systems, among others. Monitoring takes on a special role in WMNs given that, generally, there are nodes in WMNs that are not controlled or do not belong to the same administration domain. Yet, these nodes are part of the network and of the packet forwarding process. For example, in rescue scenarios, a node could filter and drop packets that are not of its own interest in order to save resources. If such a node is placed between a rescue team and a victim, for example, and the rescue team does not receive the emergency messages from the victim, it can lead to a fatality. It does not mean that the administrator of this node originally had such a malicious intention, but packet filtering in this context can lead to dramatic situations. An intrusion detection system can save a life in this scenario, if for example, it keeps the rescue teams informed of which nodes of the network are working correctly and which are not. The rescuers can go to that area where the communication is broken but people are likely to be there waiting for help. In vehicular communications, for example, cars keep communication between them and to infrastructure nodes along a highway. A malicious infrastructure node can provide false information leading to unnecessary bottlenecks, delays, or to accidents in the worst case. Intrusion detection for this application is also of utmost importance.

Intrusion detection is a necessary element of operation security. Measures provided for preventing attackers, such as encryption and authentication, can be applied to WMNs to attempt to reduce intrusions, but they do not eliminate them. For example, using encryption and authentication is not sufficient against attackers which have direct access to the compromised node, including its private keys. The security research history evidences that there exist always methods to break into networks, despite of the number of countermeasures employed. Intrusion detection systems can be considered as an important and complementary wall of defense [11].

Inspired by the distributed nature of WMNs, a common approach provided by the research community to address intrusion detection is the distributed network monitoring [12]. This implies the deployment of sensors on many or all nodes of a multihop network [13]. The collected status information is then either processed in a distributed fashion or collected and processed by a central monitoring unit [14, 15]. While this approach is very promising in theory, experimental results show that the overhead generated on the nodes is quite high and often prohibitive [16, 17]. This represents an important challenge to overcome if we consider that nodes are often cheap, resource-limited devices. Hence, we derive our first research question: *is there a practical design alternative to the distributed deployment of intrusion detection sensors in wireless multihop networks?*

A second notable challenge is the limitation of the passive eavesdropping for detection of attacks. Intrusion Detection Systems (IDSes) for wireless multihop networks presented in the literature base the detection on passive eavesdropping of data. However, sophisticated attacks can be difficult or impossible to detect by means of passive overhearing of the wireless medium [18]. Moreover, existing solutions to detect and mitigate powerful attacks require modifications to the actual protocols [15, 19–21]. The above leads us to a second open research question: *how can attacks best be detected under the assumption of resource-constrained network nodes and network protocols that cannot easily be modified to support intrusion detection?*

To find answers to the research questions, we approach intrusion detection in a different way. Instead of deploying intrusion detection systems on the nodes of a wireless multihop networks, we propose to deploy nodes into the network only for this purpose. The intrusion detection nodes are trustworthy, as well as less limited in resources (computing resources, energy resources, mobility). Ideally we would deploy one node for intrusion detection, but this might depend on the size of the network, security requirements, etc. As an analogy, we can imagine the nodes of a network as civilians of a town and our intrusion detection systems as police officers. They are deployed there to detect intrusions. Differently from distributed intrusion detection systems in literature, the intrusion detection nodes in our scheme do not detect attacks locally but detect them by looking into the nodes of the network from the outside. To this end, we propose in this dissertation the employment of an active-probing

technique. Our active probing technique uses an approach similar to classical ping or active fingerprinting, i.e., it works by sending testing packets to a host and recording/analyzing its reaction. The innovation of our dissertation is that, in contrast to fingerprinting or the classical usage of active techniques, we propose an active technique not to identify nodes, but to determine if they work according to the protocol specifications as well as to detect malicious activities. In addition, if we let the intrusion detection node be mobile, it can move through the network and examine all the nodes of a network.

1.1.2 *Taxonomy of wireless multihop networks*

A Self-Organizing Network (SON) is a concept of networks which are able to set themselves up (self configuration). The aim of self organization is for base stations to become essentially "Plug and Play" items, i.e. the configuration process should require as little manual intervention as possible. The goal of the self organization is to optimize the operational characteristics to best meet the needs of the overall network [22]. According to [23], a SON is a taxonomy of four different wireless networks: Wireless Mobile Ad hoc Networks (MANETs), Wireless Sensor Networks (WSNs), Wireless Mesh Networks, and Vehicular Ad hoc Networks (VANETs). A WMN[1] is a SON with wireless multihop communication capabilities.

1.2 GOALS AND CONTRIBUTIONS

The general goal of this dissertation is to study if active probing is practical for security purposes. If we validate our hypothesis, we aim at opening a path to using active probing for security, in particular for intrusion detection in wireless multihop networks. With this dissertation we want to establish a baseline to use active probing techniques for security to serve research purposes as well as practical applications.

The specific goals of this dissertation are:

- Design an active-probing based intrusion detection technique for wireless multihop networks. The design should be as general as possible and fulfill requirements such as efficient detection or minimal overhead.
- Identify different parameters of the active-probing process and their adjustment for practical applications.
- Evaluate the proposed active probing in a wireless multihop testbed with real attacks. Provide the community with both a solid theoretical framework on active probing as well as a concrete documented implementation.

Our contributions are:

- Our principal contribution is the active-probing-based network intrusion detection. We propose its conception, design and evaluation (Chapters 3 and 4). We evaluate the active probing and its parameters (Chapter 5).
- While keeping the focus on active-probing intrusion detection, the approach remains general. We model a temporal-selective Bayes classifier to infer the state of a network node under test which is generally applicable to other systems. For our purpose, it classifies whether a node misbehaves based on the outcome of a set of active probes. To implement this classifier, we design a recursive probe selection scheme based on the current posterior of the Bayes classifier and a prediction step. It facilitates reducing the number of active probes while maximizing the insights gained by the set of probes executed (Chapter 6).

1 We define the term WMN to generally refer to any kind of aforementioned self-organizing wireless multihop network.

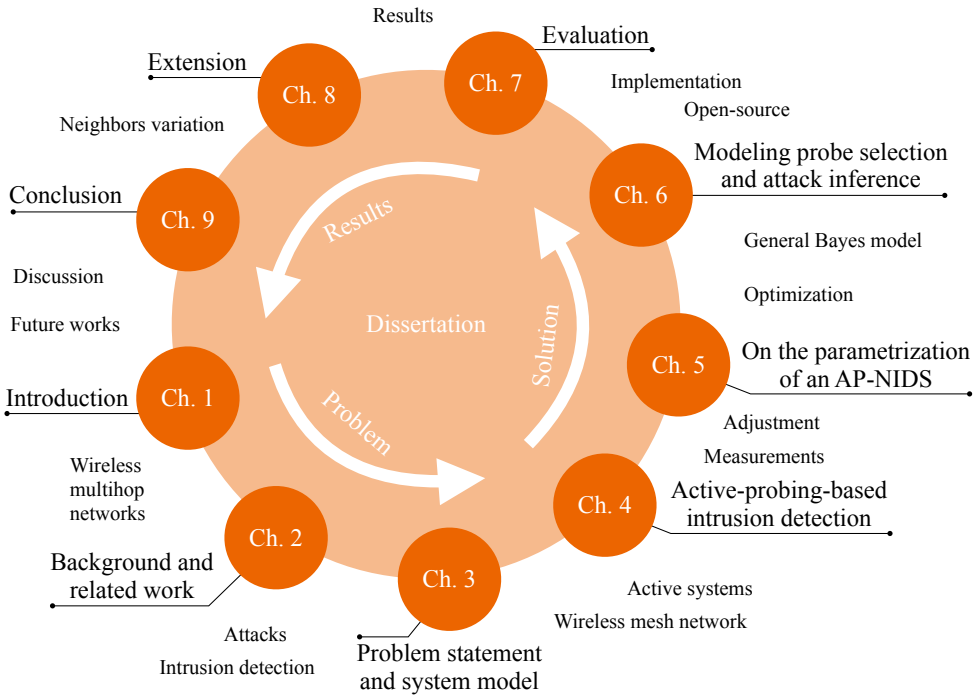


Figure 1: Dissertation outline as connected points.

- We show that other passive techniques can complement an active-probing-based intrusion detection system. We propose a lightweight metric called neighbor variation rate for anomaly detection. We study how the neighborhood varies over time and we represent it with a quantitative measurable value. We create a detection model based on this metric and we apply it for anomaly/intrusion detection (Chapter 8).
- Without losing generality, we tested our active-probing technique in an office wireless mesh testbed running the emerging mesh standard IEEE 802.11s (Chapter 7,). We provide the community with an implementation and documentation of our intrusion detection system as open source software (licensed under GPLv3) called *DogoIDS* at `https://sourceforge.net/projects/dogoids`.

1.3 DISSERTATION OUTLINE

This dissertation is structured in the following way:

In Chapter 2, BACKGROUND AND RELATED WORK, we give an introductory background on wireless multihop networks, their definition, applications, and the attacks that already exist. We also describe the different intrusion detection systems proposed in the literature.

In Chapter 3, PROBLEM STATEMENT AND SYSTEM MODEL, we introduce the problem that we address in this dissertation. We define the research questions and hypotheses. In addition, we describe the communication model and attacker model that we follow.

In Chapter 4, ACTIVE-PROBING-BASED INTRUSION DETECTION, we introduce our fundamental contribution: the active-probing-based intrusion detection technique. We explain its definitions, working principle and key concepts.

In Chapter 5, ON THE PARAMETRIZATION OF AN AP-NIDS, we focus on making the active-probing technique practical. We show that environment-dependent parameters exist, which need to be adjusted for better operation.

In Chapter 6, MODELING PROBE SELECTION AND ATTACK INFERENCE, we present a general model for attack inference and we apply it to the active-probing-based intrusion detection system to launch the so-called active probes in a probabilistic manner, minimizing the number of probes executed.

In Chapter 7, EVALUATION, we demonstrate the effectiveness of attack detection by means of active probing. We configure our intrusion evaluation system based on the lessons learned in the previous chapters and evaluate it for attack detection.

In Chapter 8, EXTENSION, we introduce an extension to intrusion detection by designing monitoring how the neighboring node rate varies over time.

Finally, Chapter 9, CONCLUSIONS, brings a close to this dissertation and raises discussion of some ideas for future directions.

The outline of the dissertation is presented graphically in Fig. 1. The bigger circle represents our dissertation, the active probing applied to security as our innovation and contribution to science. The little points are the different chapters of this dissertation. The chapters are self-contained but they follow the circle we show in the picture. Especially the evaluation is based on the chapters before.

2 BACKGROUND AND RELATED WORK

> A scientist is a mimosa when he himself has made a mistake, and a roaring lion when he discovers a mistake of others.
>
> Albert Einstein

In this chapter, we start by briefly introducing the necessary background on wireless multihop networks and wireless mesh networks. Afterwards, we go through the literature and we describe the attacks already present, and the techniques proposed to detect them.

2.1 WIRELESS MULTIHOP NETWORKS

In cellular and standard one-hop wireless networks, there is usually one wireless link between the wireless access point and the end device. In wireless multihop networks, whole or part of the path between two communicating points is composed of wireless links that forwards packets hop by hop. WMNs can extend the wireless coverage, which is useful in applications where a wired infrastructure is not possible or not desired. Each hop of a wireless multihop network decides the next hop of its incoming packets. Hence, these networks are beneficial where the participating nodes are mobile and change their position, resulting in changing the network topology. In high-loaded networks, for example, several paths can be employed to improve the network performance [24].

There are a number of scenarios for WMNs that have been recently investigated. The coverage of cellular networks can be extend by employing multihop technology. Wireless mesh networks have been applied in order to provide Internet access where wired infrastructures are not cost-effective. The mesh nodes form a multihop backbone infrastructure, which is usually fixed at users' home. Wireless mesh networks can be applied for community and enterprise networking. Wireless mesh networks can employ different technologies and routing protocols, such as the IEEE 802.11s. There exist vehicular multihop networks. These networks are applied to cars in order to exchange live information in a car-to-car way. This information can be, for example, the state of a nearer highway, or accidents. Another emerging technology for wireless multihop communications are the wireless sensor networks. These networks usually exchange data from distributed sensors with low resources to a sink, which use the data for computation. The sensors can be employed for diverse applications, for example, to perform measurements in pipelines [24, 25].

2.1.1 *Wireless mesh networks - the IEEE 802.11s standard*

In general, the active-probing mechanism proposed in this dissertation and its different respective studies are general and applicable to WMNs. However, for the sake of the validation of this work and as an example throughout the different chapters, we will often refer to intrusion detection applied to the concrete setting of a wireless mesh network running the IEEE 802.11s standard. For this reason, in this section, we describe the technical background of the IEEE 802.11s standard.

Overview

In contrast to the IEEE 802.11 [26], the standard IEEE 802.11s [27] incorporates a new kind of Basic Service Set (BSS) called Mesh Basic Service Set (MBSS) [28]. The so-called Mesh

Station (MSTA) participates in the formation of a MBSS, path selection, and any mesh-specific functionalities. The non-mesh stations do not communicate with mesh stations, and the interconnection is possible by coupling the MBSS to the Distribution System (DS) through mesh gates [27].

Mesh creation

A mesh is created dynamically between the mesh stations. They perform two variants of station discovery: passive or active scan. In a passive scan, the MSTAs limit their discovery to just observe the beacon frames, which are periodically transmitted by other mesh stations in the MBSS. In an active scan, the MSTAs send probe frames, which are answered by probe response frames by other MSTAs. Beacon frames further serve synchronization and power management tasks.

Mesh peering

A mesh station creates and maintains peer links with its neighbor nodes, which are required for direct communication. A neighbor can be found by means of passive or active scanning. Mesh peering is established between neighbors according to the Mesh Peering Management Protocol (MPM) or the Authenticated Mesh Peering Exchange (AMPE). This last one is employed if authentication is required. Mesh stations do not transmit frames other than the ones used for candidate peer mesh station discovery to a neighboring MSTA until a mesh peering has been established with the mesh station. A MSTA can establish a mesh peering with multiple neighbor MSTAs [27].

Path selection

IEEE 802.11s defines a mandatory path selection protocol: a hybrid (proactive/reactive) protocol named Hybrid Wireless Mesh Protocol (HWMP). It is inspired by the Ad hoc On-Demand Distance Vector (AODV) routing protocol [29] adapted for MAC address-based path selection and link metric awareness, and it is completely defined in the standard [27]. HWMP can be configured to operate in two modes: on-demand path selection mode and proactive tree building mode. The on-demand mode is appropriate to establish a path between MSTAs in a peer-to-peer basis. In proactive mode, a tree-based topology is calculated once an MSTA announces itself as a root. If great portions of network traffic is always forwarded to specific nodes, a tree-based protocol can improve the efficiency of the path selection [30].

Definition 1. *We define the term routing as the mechanism for path (also named route) selection/discovery, establishment and maintenance. A routing packet or control packet, for example, is a packet employed for such purpose.*

Security

The mesh stations use the Simultaneous Authentication of Equals (SAE) algorithm for peer authentication. This algorithm provides two mesh stations with a pairwise master key that they use to mutually authenticate themselves and encrypt their frames. SAE does not rely on a keying hierarchy like traditional 802.11 encryption, and it implements a distributed approach that both mesh stations may initiate simultaneously. Each link is secured due to the pairwise encryption. As a consequence, 802.11s does not provide end-to-end encryption. For broadcast communication, a broadcast traffic key exists. Peers share this key with all neighbors and update it with every new peer [28].

2.2 ATTACKS IN WIRELESS MULTIHOP NETWORKS

In this section we collect and summarize the attacks for WMNs found throughout the literature. Out goal is not only to make a mere list of attacks, but also to classify them and later analyze

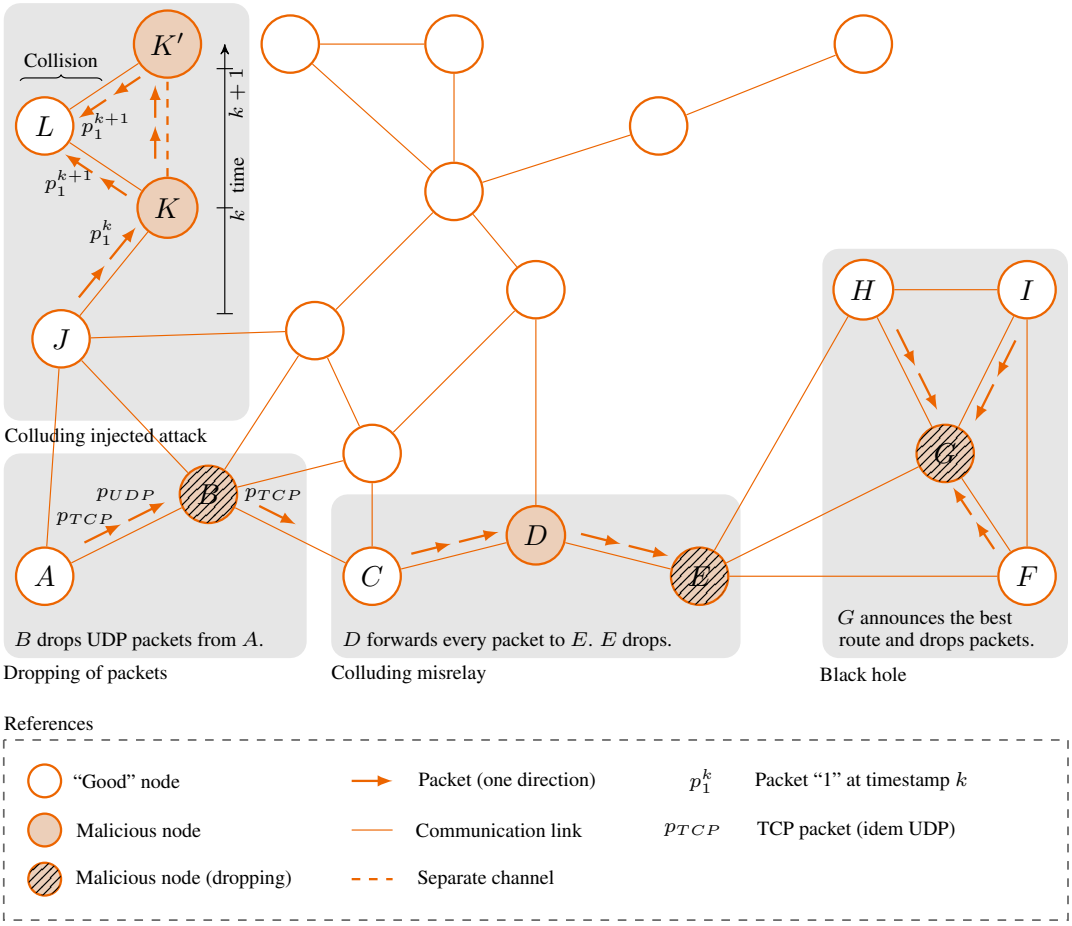


Figure 2: Conceptual description of the working principle/deployment of some attacks in WMNs.

against which kind of attacks our IDS approach performs well. First, we list and describe the different attacks. For the description of the attacks we refer to a hypothetical *malicious node*, i.e. a node launching the attacks. We will employ the singular case, however, the definitions are valid also if there is more than one malicious node.

Definition 2. *We employ the term malicious node for a compromised non-malicious node (a node that was not malicious before being compromised). We call the non-compromised or non-malicious nodes as good nodes, legitimate nodes, or non-malicious nodes.*

Definition 3. *The term neighboring node (of another) refers to a node which is one-hop away (also called next-hop node or immediate neighbor).*

2.2.1 *Insider and outsider attacks*

Network attacks can be classified into insider and outsider attacks. In general, insider attackers have access to network resources by means of having the cryptographic keys and being authenticated on the network and their presence is harder to detect [31–34], thus making detection more challenging. Many of the attacks that exist in the literature can be prevented by using cryptographic protocols to secure the network traffic (effectively preventing outsider attacks). The attacks that remain for outsider attackers are mostly performed at the physical layer, such as the jamming attack. However, preventing attacks on the physical layer requires techniques which are outside the scope of the detection technique that we propose in this dissertation.

Black hole attack

In a black hole attack, a malicious node sends fake routing information, claiming that it has the best routes to all destinations and causes other good nodes to route data packets through the malicious one. The malicious node (selectively) drops the receiving packets [20, 35–38]. In Fig. 2, for example, the malicious node G announces to its neighbors it has the best route to any destination. The neighboring nodes E, F, I and H send their packets through G and G drops them.

Colluding injected attack

The colluding injected attack is a complex attack that requires two previous phases before being launched. In the first phase, called node replication phase, the adversary compromises an arbitrary node and then injects a replication node of the compromised node in the network. In a second phase, or injection phase, the attacker inserts another node into the network. This node colludes with the replicated node with the goal of disrupting some other node's communication. Both colluding nodes originate an intentional collision of every packet transmitted to the attacked node. If a watchdog system is present in the network, it will note that the attacked node does not forward packets as expected. The watchdog can, for example, report the attacked node as malicious and exclude it from the network [39].

Colluding misrelay attack

In this attack, multiple malicious nodes work in collusion to modify or drop routing and/or data packets to disrupt forwarding/routing operation in a WMN. For example, in Fig. 2 the first malicious node D forwards packets to the second malicious node E to avoid being detected by node C. The second malicious node E drops the received packets [20, 40].

Cooperative black hole attack

Similar to the colluding misrelay attack, a cooperative black hole attack occurs when two or more malicious nodes collude. The difference to the colluding misrelay is that the colluding nodes always reply by announcing one of them as the best route to all destinations, and then dropping the receiving packets [41, 42].

Fabrication and modification attack

A malicious node injects routing packets with false routing information. Similarly, the modification attack modifies routing information of received routing packets and retransmits them [36, 43].

Fuzzing attack

In this attack a malicious node injects poorly formed (invalid) packets into the network. The content of the packets can be completely random or only some fields. They can also be genuine packets cut short. The packets can belong to data and/or control packets [44].

Jellyfish attack

Jellyfish attacks work on WMNs that use protocols with congestion control techniques, such as the Transmission Control Protocol (TCP), in the transport layer. Packets can be maliciously dropped (Jellyfish Periodic Dropping Attack), reordered (Jellyfish Reordering Attack), or delayed (Jellyfish Delay Variance Attack) to adversely affect network throughput [38]. Refer to [37] for a detailed description of this attack and its variants.

Man-in-the-middle attack

In a man-in-the-middle attack, the attacker places itself between two communicating nodes without their consent. In this way, the attacker can eavesdrop the communication and insert, modify or delete network traffic. A man-in-the-middle attack can be used to steal web sessions, for example, or to bypass authentication mechanisms. It can be also employed to traffic analysis and obtain information from the legitimate users [45].

Resource depletion attack

A malicious node sends out unnecessary routing control messages, which can congest channels, use up battery power, and fill up storage space. Alternatively, a malicious node could also establish communications with other nodes (or colluding nodes) and send unnecessary or fake data repeatedly [38]. Examples of resource depletion attacks include the "vampire attacks", introduced by Vasserman and Hopper [46]. One type of vampire attack is the carousel attack, where the attacker crafts packets with routing loops introduced. It exploits the weak message headers verification of routing protocols, which allows a single packet to traverse the same nodes again. Another vampire attack is the stretch attack. An attacker create long paths, which might cross through every node in the network. The attacker can increase the lengths of the packets, which might cause the long packets to travel to a number of nodes independently of the shortest path to the destination [46].

Routing loop attack

In routing loop attack, a malicious node receives a message, updates it, and sends it back to one of the previous forwarders (or the source) even if there is a better node in its routing table that is available to be the next forwarder according to the routing strategy. The aim of this attack is to delay or prevent the delivery of a message [43, 47].

Selective dropping of packets attack

A malicious node exploits the cooperative nature of WMNs and it drops packets that are supposed to be forwarded to neighboring nodes [48]. The packets can be dropped in a random drop mode, where any packet is dropped with a given probability, and/or also in a selective mode, where a malicious node only drops certain types of packets (services) with a given probability. In selective mode, the attacker can drop a small number of critical packets, for example, control packets. This drop might seriously degrade the network performance [36, 38, 49, 50]. If the dropping probability is 1 for every kind of service, this attack is simply called "dropping of packets". A conceptual representation of this attack is depicted in Fig. 2, where node B drops packets incoming from node A.

Selfish attack

The selfish behavior attack occurs when a malicious node does not participate either in the route discovery mechanism, or in the packet forwarding, or also not in either of them, and it only does it when the malicious node itself needs to communicate. Generally, the purpose of this attack is to conserve energy [38, 51].

Wormhole attack

The wormhole attack, as introduced by Hu et al. [52], happens when an attacker tunnels packets from one point to another point of the network using a high quality out-of-band link. The tunneling causes that the nodes find routes to far away destinations only traveling throw the tunnel [38]. The malicious node can be set either in hidden mode or in participation mode, as defined by Khabbazian et al. [53]. In the first one, the attacker is an external node, invisible for the legitimate nodes. The tunneling occurs at the physical layer, where the bits are captured in one region and forwarded in other; in the second mode, the attacker is an

Table 1: Classification of common attacks in WMNs

Attack name	Attack goal	Target	Type
Dropping of packets†	Denial of Service - Performance degradation	Data	I
		Routing	I
Jellyfish	Performance degradation	Data	I
Black hole†	Denial of Service	Routing	II
Selective dropping of packets†	Denial of Service - Performance degradation	Data	II
		Routing	II
Fabrication and Modification	Degrade the quality of service by creating misrouting situations	Data	II
		Routing	II
Fuzzing	Denial of Service by leading to crash or to excessive resource consumption	Data	II
		Routing	II
Resource depletion	Increase resource consumption leading nodes to their limits of processing and memory capacity - Increase link utilization	Data	II
		Routing	II
Selfish	Performance degradation by not collaborating during the routing process	Routing	II
Cooperative black hole	Denial of Service	Data	III
Routing loop	Denial of Service - Increase of link utilization	Data	III
Colluding misrelay†	Denial of Service	Data	III
Colluding injected attack	Deny communication to a victim node	Data	III
Man-in-the-middle	Control the communication between nodes affecting their integrity and/or privacy - Denial of certain services by selective dropping	Data	III
Wormhole	Performance degradation of particular services, and compromised privacy	Routing	III

† Attack implemented during the evaluation of this dissertation.

insider node which is likely to be authenticated by the other nodes and which participates in the path discovery and packet forwarding process [20, 38, 45, 52].

2.2.2 *Taxonomy*

We propose a classification of the different attacks in WMNs according to how they can be detected. We create this classification in this sense because it helps us to understand how these attacks are linked, from the perspective of an intrusion detection system. First, we define four properties as follows:

- *Local:* The attack can be detected with the information gathered locally, including observable traffic from the immediate neighbors.
- *Distributed:* Information from nodes beyond the neighboring nodes is needed to detect the attack.
- *On-the-fly:* No previous data is needed to detect the attack.
- *Correlation:* The attack is detected only after gathering and correlating a certain amount of data.

Based on these properties, we define the following three types of attacks with increasing detection complexity:

- *Type I:* Local, On-the-fly
- *Type II:* Local, Correlation
- *Type III:* Distributed, Correlation

Note that we skipped the category "Distributed, On-the-fly" because it is inconsistent. If the IDS needs distributed information in order to detect the attack, it cannot be detected "on-the-fly" as defined above. In Table 1 we summarize the classification of common attacks. We distinguish the attacks that affect the data packets and those which affect the routing packets.

For the evaluation of this dissertation in the following chapters, we choose one representative attack of each type and deploy them in our testbed. For attack Type I, we choose the *(selective) dropping of packets* as a typical example of an attack that could be detected without the need for complex analysis. This attack is very important, in spite of its low complexity. It is relatively easy to implement, which make it attractive to any attacker with a basic knowledge of computer systems. This attack can cause serious damage to the network (e.g. by dropping control packets). For Type II, we use the *black hole* attack, which is very popular in the literature related to multihop networks, and its detection implies collecting information from neighbor nodes in several steps. For Type III, we choose the *colluding misrelay* attack because the detection requires the gathering of information from different parts of the network.

2.3 INTRUSION DETECTION SYSTEMS

Next we describe relevant different approaches for intrusion detection in WMNs presented in the literature.

Acknowledgment-based IDS

Shakshuki et al. presented EAACK [54], an intrusion detection system that is based on secure acknowledgments and misbehavior monitoring. It basically employs an end-to-end acknowledgment scheme, i.e. it relies on packet acknowledgments in order to detect attacks or misbehavior.

Distributed and cooperative IDS

An IDS architecture for MANETs first appeared in the work of Zhang et al. [11, 13] and continued by Huang and Lee [14]. The proposed architecture is based on randomly setting some nodes on the network as intrusion detectors, building a cluster of sensors which collaboratively works towards global anomaly detection. DEMEM is another distributed and cooperative IDS model presented by Tseng et al. [55]. The nodes exchange messages for more than one-hop detection. The messages contain critical information called evidence and are used to validate the protocol specification. This model cannot detect collaborative attacks. MASID [56] is yet another very similar architecture for distributed intrusion detection in MANETs.

Cluster-based IDS

Kachirski and Guha [57] present a hierarchical and distributed IDS where the network is divided into clusters. Each node performs local intrusion detection and only the cluster head nodes perform network intrusion detection. Clusters are chosen by vote based on connectivity. The performance of intrusion detection strongly changes depending on the hop distance to a cluster node.

Cross layer IDS

Shrestha et al. [58] present an IDS for MANETs that employs a cross layer mechanism for attack inference. A cross-layer algorithm collects traces from the different layers of the nodes that it is supervising In addition, distributed intrusion detection is achieved by exchanging information with neighboring nodes that run the IDS as well.

Specification-based IDS

Tseng et al. [15] present a model for intrusion detection for AODV that detects the effects of the intrusions as run-time violations of the protocol specifications. The detection is performed on strategically selected monitoring nodes which cover all the nodes of the network. It is assumed that the monitors know the IP and MAC addresses of all the nodes of the network, and the MAC addresses cannot be forged. Moreover, the AODV protocol needs to be modified by adding a new field. Vigna et al. [17] describe AODVSTAT, a tool based on a misuse detection technique for the AODV protocol, which is supposed to be deployed on a set of nodes of the network. It can detect intruders locally (i.e. within the immediate neighborhood of the node), as well as in a distributed cooperative manner.

Learning technique

A different approach to intrusion detection based on a grammatical evolution technique is presented by Şen and Clark [59], in which the intrusion detection programs are evolved and distributed to each node on the network for each type of attack.

Lightweight IDS

Hugelshofer et al. [16] introduce a lightweight IDS called OpenLIDS. They considered the low resources of the mesh nodes and emphasize detecting the attacks efficiently, with low resource consumption. Their experiments showed that with lightweight detection metrics they were able to detect several threats, though for more complex ones, the resource consumption was not significantly different compared to other conventional IDSes.

Statistics-based IDS

Anjum and Talpade [60] presented LiPaD, a practical approach to detect packet dropping attacks in MANETs. The IDS is deployed on every node and the employed algorithm is simple and lightweight. The nodes send reports to certain coordinator, which initiates attack detection. The network bandwidth can be quite large because each node sends reports of each data flow.

2.3.1 *Comparison of the different intrusion detection systems*

In Table 2 we compare the different intrusion detection systems presented in the literature and our active-probing-based network IDS proposed in this dissertation. We compare if the different intrusion detection systems are able to detect the three different kinds of attacks (described in Section 2.2.2). Performing such a comparison is not easy, since the IDSes are all designed, described, and evaluated differently. Based on the literature work, we reasoned whether or not the IDSes are capable of detecting of detecting the different kind of attacks. Of course, we understand that in some cases this might be arguable, however, in most cases it is quite clear what the IDSes can and cannot detect.

We classify the intrusion detection systems according to their detection technique employed. We follow the classification employed in [66]:

Table 2: Comparison of the different intrusion detection systems proposed in the literature

Property	Huang and Lee [14]	DEMEM [55]	MASID [56]	Shrestha et al. [58]	EAACK [54]	Kachirski and Guha [57]	AODVSTAT [17]	Tseng et al. [15]	Şen and Clark [59]	OpenLIDS [16]	LiPaD [60]	Sterne et al. [61]	Nadeem and Howarth [62]	Cretu et al. [63]	Mitrokotsa et al. [64]	Orset et al. [65]	*IDS of this dissertation*	Observations
	Intrusion detection system																	
Type I	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
Type II	•	•	•	•		•	•	•				•					•	
Type III	•						•										•	
Testbed (T) / Simulation (S)	S	S	-†	S	S	-†	T	-†	S	T	S	-†	S	S	S	-†	T	
All nodes have IDS	•‡	•	•	•	•	•‡	•	•	•	•	•	•	•	•	•	•		
Exchange messages between nodes	•	•	•	•		•	•					•	•	•	•		•⋆	
Detection technique¶	ABID	SBID	Hybrid	ABID	SBID	ABID	KBID	SBID	KBID	ABID	ABID	ABID	ABID	ABID	ABID	SBID	Hybrid§	
																		Complex cluster formation
																		Every node validates routing packets
																		Strong based on local detection
																		Digital signatures
																		Complex cluster formation
																		Thought for AODV
																		All MAC and IP addresses are known
																		Efficient for simple attacks
																		Synchronization required; only detects packet dropping
																		Complex hierarchical cluster formation
																		Distribution of anomaly models
																		Exchange of maps with security status

† The IDS was presented as an idea/model, without evaluation.
‡ The IDS can be deployed to a subset of nodes.
⋆ Not always necessary. Exchange between IDS nodes.
¶ Notation from [66]; ABID = anomaly-based IDS, KBID = knowledge-based IDS, SBID = specification-based IDS, Hybrid = a mixture of all/some of these.
§ KBID and SBID.

- *Anomaly-Based Intrusion Detection (ABID): "Also known as behavior-based intrusion detection, it flags as anomalous observed activities that deviate significantly from the normal profile"* [66].
- *Knowledge-Based Intrusion Detection (KBID): "Also known as misuse intrusion detection, knowledge-based IDS maintain a knowledge base that contains signatures or patterns of well-known attacks and looks for these patterns in an attempt to detect them"* [66].
- *Specification-Based Intrusion Detection (SBID): "Specification-based IDSes first explicitly define specifications as a set of constraints. They then use these specifications to monitor the routing protocol operations or network layer operations to detect attacks in the network"* [66].
- *Hybrid:* A hybrid of the various systems. For example, an IDS can be both anomaly-based and knowledge-based. It has the advantage of correctly detecting attacks that are already known (knowledge) and at the same time being able to detect new attacks based on deviations of the normal activity (anomalies). However, configuring such an IDS is more complex because of having to set up both components.

2.4 SUMMARY

In this chapter, we provided the background on wireless multihop networks necessary to understand the work presented in this dissertation. Additionally, we investigated and classified the attacks that already exist. We surveyed related works and described the state-of-the-art for intrusion detection systems for wireless multihop networks. We described and classified the different approaches to intrusion detection. The background provided in this chapter is necessary to understand why we decided to investigate intrusion detection, why and at which point the actual system fails, and how different our approach is from these others. In this chapter, we highlighted three conclusions. First, few intrusion detection systems are able to detect Type III attacks. Secondly, there are few practical implementations of intrusion detection systems for WMNs. Thirdly, the proposed intrusion detection systems in the literature have to be implemented in all the network nodes.

3 PROBLEM STATEMENT AND SYSTEM MODEL

> Research is what I'm doing when I don't know what I'm doing.
>
> Wernher von Braun

> In school, we're rewarded for having the answer, not for asking a good question.
>
> Richard Saul Wurman

In this chapter, we concretely define the problem statement of this dissertation, our fundamental research question and hypothesis. We also define the communication model and attacker model that we employ.

3.1 PROBLEM STATEMENT

Wireless multihop networks can be instrumental for several scenarios, such as search and rescue in disaster areas, thus potentially helping to save lives. For this reason, detecting the attacks that can disrupt the network operation can be considered of utmost importance. By means of an extensive literature study (see Section 2.3), we found severe limitations in the field of intrusion detection for WMNs in general. Purely centralized intrusion detection systems are ill suited for WMNs, because WMNs miss a clear line of defense due to their distributed and wireless nature. Moreover, in a number of application scenarios, WMNs lack a centralized management authority. Hence, the most studied approach to intrusion detection in WMNs is the distributed IDS [12], which consists of deploying detection sensors in all or part of the nodes of the network [13]. The attack detection is performed by means of passive eavesdropping of the wireless medium. Many different approaches following this distributed detection paradigm have been proposed in the literature, although many of them have not actually been tested in practical implementations but in computer simulated environments. Where practical validation was performed, the results showed that the overhead generated on the nodes, in terms of memory and CPU consumption, is excessively high [16, 17], which makes the distributed detection principle unsuitable for networks operating on resource-constraint devices. In addition, passive eavesdropping of the medium severely limits the number of attacks that can be detected in WMNs [18]. Specification-based IDSes facilitate the detection of advanced attacks against WMNs, however they require modifications to the base routing protocols in use [15, 19–21]. In summary, intrusion detection mechanisms for wireless multihop networks proposed in the literature are scarce, and most of the proposed solutions have never been validated in practice.

In this dissertation we present a new paradigm in intrusion detection. Our working hypothesis is that active intrusion detection can overcome the problems of passive and distributed intrusion detection approaches, while being more practical than contemporary approaches to intrusion detection in WMNs.

In particular, the goal is to allow for 1) efficient detection of 2) a wide class of attacks (including advanced attacks). Moreover, the proposed solution needs to be 3) practical for heterogeneous and constrained environments as well as for legacy protocols.

Our proposed detection method does not need the modification of the routing protocols. The objective of our intrusion detection technique is to detect certain classes of insider attacks

(see Section 2.2.1). Although the method we propose is general and applicable to wireless multihop networks in general (different degrees of mobility, routing protocols, transmission capacities, among other properties), for the sake of the validation in practical environments, we have chosen the class of *wireless mesh networks*. Our interest in them is twofold. On the one hand, they are suited to heterogeneous environments. These networks are ad hoc, they handle high bandwidth in real-time, their nodes might be static or mobile, and energy and/or resource constraints apply. They provide a dynamic, easy to deploy and self-configurable network, for example, to serve as a backbone for other emergency response networks. On the other hand, the growing relevance of wireless mesh networks is mirrored by community efforts that aim at building organically growing mesh communities [1, 2] as well as research projects that investigate technical limitations of multihop wireless networks [3]. In addition, wireless mesh networks were recently standardized with the IEEE 802.11s mesh extension, which is quite promising given the success of more than a billion devices with built-in 802.11 capabilities in 2011, expecting to double this number by 2015 [4].

3.2 COMMUNICATION MODEL

We study the intrusion detection in wireless multihop networks. The communication model employed in this dissertation corresponds to the standard model employed in wireless multihop networks. One node of the network can communicate (send/receive packets) with other nodes if they are within its radio transmission range. If one node wants to communicate with other nodes that are not within its transmission range, the node should transmit its information to one of its neighbors, starting a chain of collaborative delivery of information. The path to a destination node is discovered in a previous step by using a routing protocol. For the sake of validation, we perform the experimentation of this dissertation using the IEEE 802.11s mesh standard.

3.3 ATTACKER MODEL

In this section we define the attacker model followed in this dissertation. Cordasco and Wetzel first defined in [67] a common attacker model to analyze secure routing protocols in MANET. We define our own attacker model inspired by [67].

3.3.1 *Capabilities of the attacker*

We assume that the attacker is capable of either introducing one or multiple malicious nodes, or of taking control of one or multiple legitimate nodes of the network (byzantine behavior). In both cases we assume the malicious nodes to be internal to the network, i.e., the node(s) belong(s) to the network.

3.3.2 *Communication*

The malicious/compromised node can freely communicate with its neighboring nodes. It can create, manipulate, send, receive, forward, and drop packets. The malicious node uses the same hardware and radio capabilities as the rest of the nodes of the network. The attacker cannot increase the transmission power or improve its sensitivity. We assume that the attacker cannot launch attacks on the physical layer (like jamming attacks). The malicious node can collude with another malicious node on the network to launch an attack, and they are able to use another communication channel.

3.3.3 *Computation*

The computation and power capacity of the attacker are limited to the resources of the node(s) that it is controlling. The attacker can only perform operations (without key material) that are feasible in a reasonable amount of time on current hardware [67].

3.4 DETECTION MODEL

The active-probing technique we present serves as a generator of audits. The audits are general, and can be applied to any detection model. In this dissertation we present the active-probing technique in operation with a network intrusion detection system. For this purpose, we build the intrusion detection system and we adopt the specification-based and the knowledge-based model (see Section 2.3.1). We decide to use these models combined because of their practicability. If, instead, we choose to use anomaly-based mode, an active-probing-based intrusion detection system would need to first transmit many probes in order to create a normal baseline that is later used to look for deviations.

3.4.1 *Definition of intrusion*

In this dissertation, intrusions are considered as any set of deliberate actions that disrupt the operation of a wireless multihop network. These actions can correspond to violations to the routing protocols in use (i.e. dropping routing packets on purpose), or can be actions that do not violate any specification but still disrupt the operation of the network (i.e. replying to have the best route to a certain destination).

3.5 SUMMARY

In this chapter, we defined the fundamental problem that we address in this dissertation. We explained why contemporary intrusion detection systems are not practical for wireless multihop networks. In the next chapter, we start describing the design of our proposed solution. We base our solution on the standard communication model of wireless multihop networks we described in this section. In further chapters, we validate our intrusion detection system by using real attacks. For this reason, this section addresses the attacker model we employ. We model the attacker as an insider attacker and we define its capabilities and limitations. In addition, we define the detection model used in our solution.

4 ACTIVE-PROBING-BASED INTRUSION DETECTION

> That's been one of my mantras — focus and simplicity. Simple can be harder than complex. You have to work hard to get your thinking clean to make it simple. But it's worth it in the end because once you get there, you can move mountains.
>
> Steve Jobs

In the previous chapters, we introduced the problem that motivates this work. We briefly described the already existing IDSes and attacks for wireless multihop networks. In this chapter, we propose a novel intrusion detection technique and we conceptually present its design and working principle [68].

4.1 INTRODUCTION

A classical definition for intrusion detection system[1] states: *"An intrusion-detection system dynamically monitors the actions taken in a given environment, and decides whether these actions are symptomatic of an attack or constitute a legitimate use of the environment"* [69]. Although active-probing-based intrusion detection is, strictly, covered by this general definition, we re-phrase it to address specifically active-probing in the context of WMNs in the following definition.

Definition 4. *An Active-Probing-Based Network Intrusion Detection System (AP-NIDS) dynamically injects probes and monitors the reaction of the nodes to the probes in a given WMN, and decides whether these reactions are symptomatic of an attack or constitute a legitimate use of the environment (adapted from [69]).*

In order to understand where exactly the active-probing-based detection is located macroscopically in the architecture of an IDS, we will refer to the terminology adopted by Debar et al. *"The term audit denotes information provided by a system concerning its inner workings and behavior"* [69]. A *system*, in this definition, refers to an information system being monitored by an intrusion detection system. In traditional IDSes (not active-probing-based), audits are composed of logging components (files, operating system logging, etc.), live network traffic, among others. These audits are monitored by a certain detector component of the IDS. An active-probing-based IDS also features a detector component that monitor the audits of the system being monitored. The fundamental difference lies in how the audits are generated. In traditional IDSes, the audits are generated by the system being monitored inherently from the IDS. In an AP-NIDS, on the other hand, the audits are generated by the system being monitored *as a consequence of the probes previously generated by the AP-NIDS.* We extracted a depiction of the taxonomy of IDSes according to their functional and non-functional characteristics from [69] and we included the new *audit method* proposed in this dissertation. This new taxonomy is shown in Fig. 3.

4.2 WORKING PRINCIPLE

Conceptually, the goal of the active probing technique is to uncover malicious node behavior by the creation and transmission of packets from an intrusion detection node to a node or

1 We employ the term *intrusion detection system* in a generic manner and we define and employ the term *active-probing-based network intrusion detection system* to refer in particular to network IDSes with active-probing techniques.

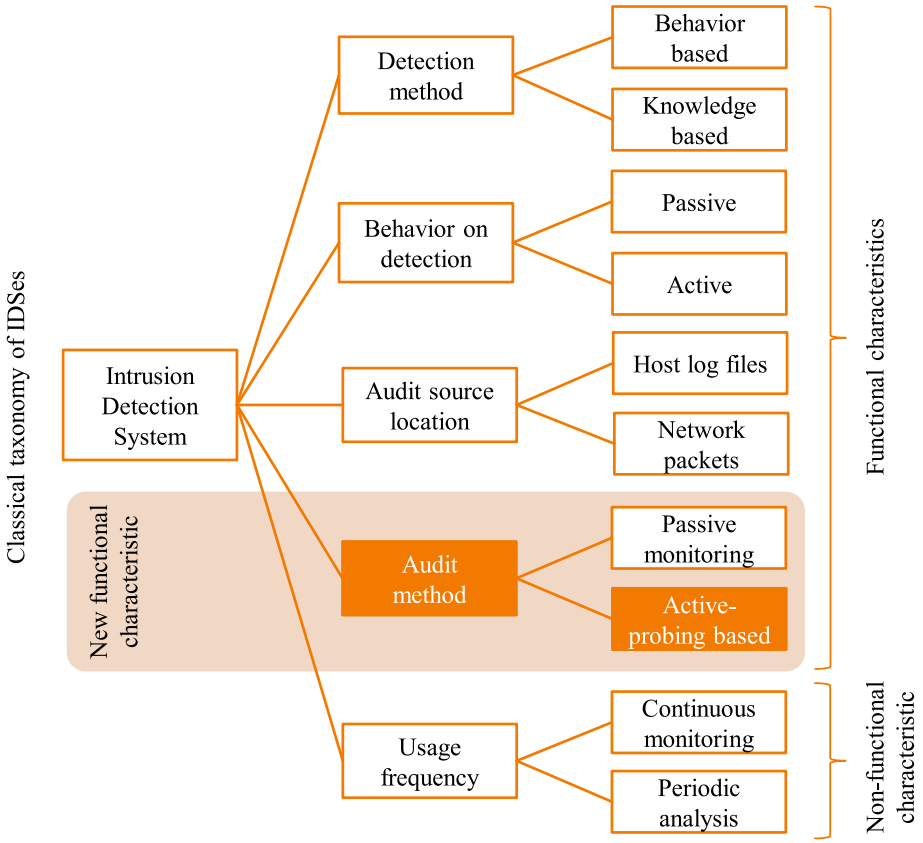


Figure 3: Classical taxonomy of intrusion detection systems from [69] and the new audit method proposed in this work.

group of nodes under test. In the most basic configuration, the active intrusion detection is composed of at least two parties: a node serving as an intrusion detection system (the node that sends the packets) and a node being tested, called *target node*. The packets transmitted by the intrusion detection node for intrusion detection are called *testing packets*.

The testing packets are transmitted within the context of a *probe*.

Definition 5. *A probe is the set of steps and testing packets involved to detect one particular attack.*

For example, a probe can be created to detect "selective dropping of HTTPS packets". To do so, it might send several different testing packets. Although a probe is well-defined to detect one particular attack, a probe may also hold information about attacks it is not designed for.

After the testing packets are sent to the target node, an AP-NIDS gathers the traffic generated by this node (if any) and analyzes it. The relationship between the basic components of active probing is shown in Fig. 4. A conceptual representation is shown in Fig. 5. In the figure, the node F is the target node, and p_1 and p_2 are testing packets. As a simple example, the AP-NIDS could test if the node F drops the testing packets p_1 and p_2. The intrusion detection node can be a static node fixed at some position, or it can be a mobile platform. By being mobile it offers two major advantages. First, it can cover a larger area of the network without the necessity to deploy more intrusion detection nodes. Secondly, it can strategically locate itself to detect certain kind of attacks. The AP-NIDS depicted in in Fig. 5 has a link to the node A and F. The figure could be seen as a snapshot in time. If the intrusion detection node moves, at a future time it will not have a connection to A and F but to the rest of the nodes at the right, for example.

The testing packets are assumed to be indistinguishable from regular packets in the network to conceal the AP-NIDS from malicious nodes. The reaction of the nodes to these packets depends on the packets themselves and on the chosen protocol. For example, data packets

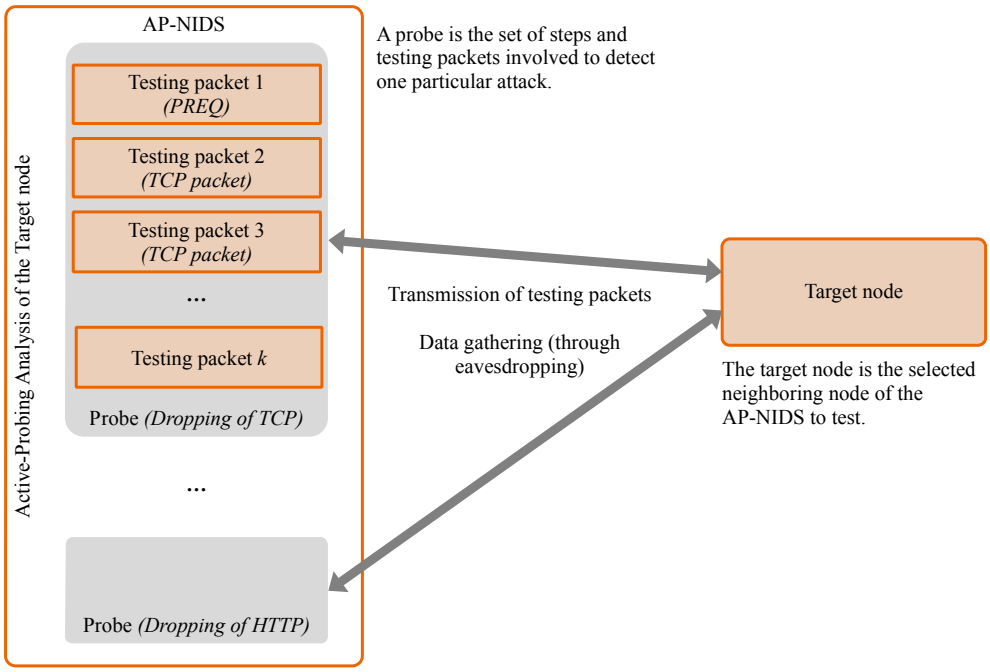


Figure 4: Depiction of the basic components of active probing. *PREQ*, *TCP packet* and *Dropping of TCP/HTTP* are only for the sake of example.

might need to be forwarded to neighboring nodes while path discovery packets are replied to according to the specification of the protocol. In order to accomplish the active-probing-based intrusion detection, the AP-NIDS performs three general steps: 1) Selection of the target node, 2) transmission of testing packets, and 3) compilation and analysis of the data generated by the target node after the active probing (if any). Data is only gathered by eavesdropping. No explicit audit information is required from the target node. After completing the above steps, the AP-NIDS infers that: 1) Either the target node behaved as expected and no attack is detected, 2) an attack was detected and classified, or 3) the AP-NIDS cannot assess the target node sufficiently.

4.2.1 *Mobility of the IDS node*

The mobility of the IDS nodes in our approach addresses the issue of network wide coverage, without requiring the deployment of IDS components on every node in the network. Autonomous mobile robots are used in a wide range of industrial applications, such as safety inspection of complex infrastructures like pipes, turbines in power plants [70, 71], or detection of gas leaks in chemical plants [72]. We take inspiration from them and we consider that we can also benefit from mobile systems for intrusion detection in wireless mesh networks, not only limited to industrial deployments. In addition, we want to detect sophisticated threats in which the position of the IDS node plays a fundamental role. For example, in Fig. 2 (on page 9) nodes D and E are performing one kind of a *colluding misrelay attack*, in which node D forwards all packets to E, and E drops them. A fixed IDS only sees that D forwarded the packets. In the case of a colluding attack, this is the expected behavior and without further information on the behavior of node E, this attack cannot be detected. If the IDS is mobile it can discover this malicious behavior by strategically placing itself between D and E.

Mobility and scope of this dissertation

In this dissertation we propose the employment of an active-probing-based network intrusion detection. We analyze that it is possible to deploy an AP-NIDS on a fixed or mobile platform,

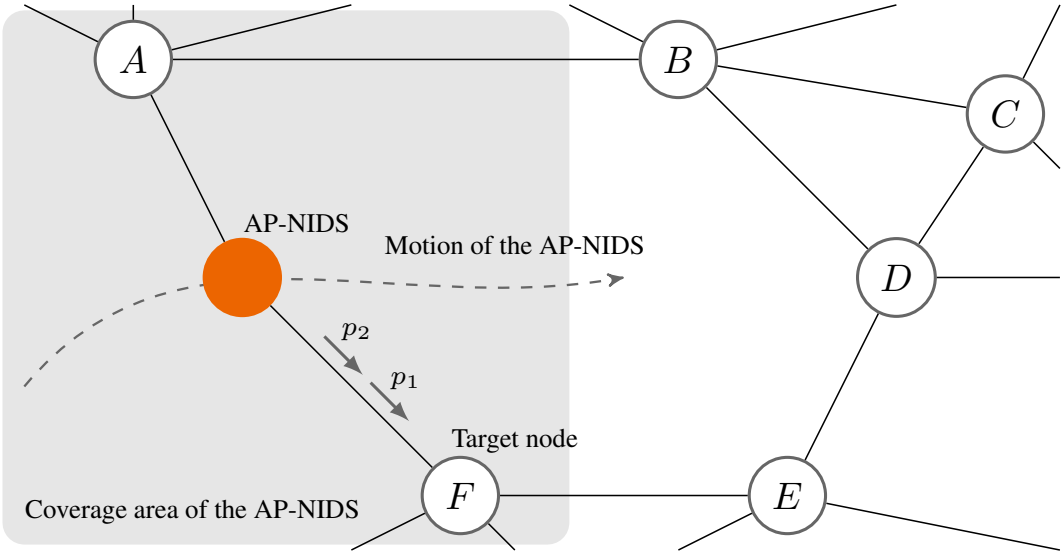


Figure 5: Conceptual representation of the active-probing-based intrusion detection scheme.

and that by being mobile it has a number of advantages. However, the focus of this work is on studying the active-probing technique per se, which is the fundamental part of an AP-NIDS. A detailed study of the mobility of an IDS is out of the scope of this dissertation and we will leave it for future research.

4.3 ACTIVE PROBING IN OTHER DOMAINS

It is interesting to investigate if active-probing techniques are successfully employed in domains other than security; in this section, we review the literature, briefly showing similar active techniques in fields other than just network or computer-related applications.

Ping and traceroute

The *ping* might be the most obvious example when it comes to active measurements. Mike Muuse created the tool in 1983, and named it in analogy to the sound that a sonar makes, inspired by the whole principle of echo-location [73]. In fact, a sonar is another example of an active-probing technique. *Traceroute* is another trivial example of an active measurement tool.

Network performance measuring

One of the most popular usages of active techniques in computer networks could possibly be to measure the performance of the network links. We cite some examples. In [74], active measurements are used for analyzing the performance of multi-radio nodes. The authors build a setup where UDP streams are sent at different packet rates with different packet sizes. The authors of [75] show how active measurements in IP backbones are used to provide a comprehensive view of the network performance and help to resolve problems in an operational setting. Fok et al. use in [76] active measurements for network faults diagnosis. For example, for measuring the round-trip time they transmit probes consisting of two TCP data packets carrying an HTTP request. These packets induce a response of two TCP data packets carrying HTTP responses from the web server. Luckie presented Scamper [77], a packet-prober tool that facilitates the conduction of large-scale measurements. In his work, the author shows the tool with different active measurement techniques such as ping, traceroute, alias resolution, and many others.

Network/traffic characterization

Actives techniques are also popularly employed for characterizing networks. Hu and Steenkiste [78] propose active techniques for characterizing the available bandwidth of a network. In [79], an active technique is presented for Internet traffic characterization. They insert a train of packets at one edge of the network and measure its dispersion. The authors of [80] use end-to-end active measurements to discover key network parameters in 3G mobile networks. A practical and successful example of probes for network characterization is the Atlas experiment, carried out by the The Réseaux IP Européens Network Coordination Centre (RIPE NCC). RIPE freely ships probing devices to individual volunteers which connect them to the Internet. The goal of the RIPE Atlas is to create a global network of probes that measure Internet connectivity and reachability in real time [81].

Network security

Not much previous work uses some kind of active technique for security purposes. Shankar and Paxson developed a technique called Active Mapping to be used with network intrusion detection system. The idea is to provide a Network Intrusion Detection System (NIDS) with information about the intranet that it monitors. Active Mapping creates a profile with the characteristics of the monitored hosts and of their network topology [82].

Network topology mapping

Another notable use of active-probing techniques is for Internet/network topology mapping. By means of probes, topology generators aim at building a graph of a certain network or even of the Internet. These graphs, however, are just approximations [83, 84].

Radar

A radar is an example of a popular active tool which is not related to network computers. A radar operates by actively transmitting electromagnetic signals and detecting the echos from the targets withing its coverage area [85].

Positron emission tomography

To continue with active-probing-based utilities that are not related to network computers, we mention the well-known positron emission tomography used in medicine [86]. This is also a measurement that involves an active component: A scanner measures the gamma rays emitted by positron-emitting radionuclide introduced into the body.

Summary

Our work has been inspired by the applications presented in this chapter. Active probing applications have already been used with much success. If we look at the network characterization, which is a closer example to our system, evidence is found of interesting highlights to consider for our work. The RIPE Atlas probing, for example, is an invasive measuring system, where the user accepts to share a part of the Internet bandwidth at home for active probing. The probes, however, are lightweight and the overhead virtually does not affect the normal usage of an Internet connection. According to their documentation, the overhead of an IPv4-only probe is approximately 4 Kbit/s [81]. About 7000 probes are connected at the present time. We take this example as a case of a rather successful active probing measurement. With our active probing IDS we aim to build a system that has similar requirements. It should perform active probing and it should be lightweight in terms of network overhead, while detecting attacks correctly.

4.4 GOALS AND METRICS

In this section we detail the goals of an active-probing-based network intrusion detection system. We first define the goals as they are defined in the literature for classical IDSes, especially in the work of Debar et al. in [69], and we will later complement these definitions by bringing the active-probing technique into the field. In addition, we define the metrics that we later systematically employ for the evaluation of this work throughout the different chapters.

4.4.1 *Effectiveness and efficiency*

We follow the definitions of Mishra et al. in [12]. *"Effectiveness is how to make the intrusion detection system classify malign and benign activity correctly. Efficiency is how to run the intrusion detection system in the most cost-effective manner as possible"* [12]. We define cost as the network overhead dedicated to intrusion detection. *"These two key requirements that any IDS must fulfill in essence suggest that an IDS should detect a substantial percentage of intrusions into the supervised system, while keeping the false alarm rate at an acceptable level at a lower cost"* [12].

Measurement metrics

Chapter 6 is dedicated to the effectiveness of our proposed intrusion detection system. We employ metrics such as *false positive rate* and *detection rate*. The efficiency will be indirectly measured throughout the chapters. We will measure the network overhead introduced by our AP-NIDS and discuss why it is efficient.

4.4.2 *High accuracy*

"Inaccuracy occurs when an intrusion detection system flags as anomalous or intrusive a legitimate action in the environment" [69]. This definition is orthogonal to the audit method employed, which means that there is no difference whether or not an active-probing technique is employed. According to this definition, for achieving high accuracy, the AP-NIDS should ideally flag as few legitimate actions as intrusive as possible.

Measurement metric

We employ again the false positive rate. It is a direct indicator of the accuracy of an IDS. In this work, the false positive rate is defined as the rate of attacks reported by the AP-NIDS when the target nodes are not launching attacks.

4.4.3 *High performance*

"The performance of an intrusion detection system is the rate at which audit events are processed. If the performance of the intrusion-detection system is poor, then real-time detection is not possible" [69]. There is a discrepancy here with respect to active-probing-based IDSes. Given that an AP-NIDS needs to inject traffic into the network to generate the audit events, which by design requires much more time than simply monitoring a body of data, an AP-NIDS would not be able to do a continuous real-time detection in the sense a traditional IDS does. The performance of an AP-NIDS can be evaluated as the rate at which audit events are *generated and processed*, and it cannot be compared to traditional IDSes where audits events are only processed. For achieving high performance, the AP-NIDS needs to achieve low detection times per target node.

Measurement metrics

We will employ the *detection time*, total and per node, as a metric for evaluating the performance of an AP-NIDS.

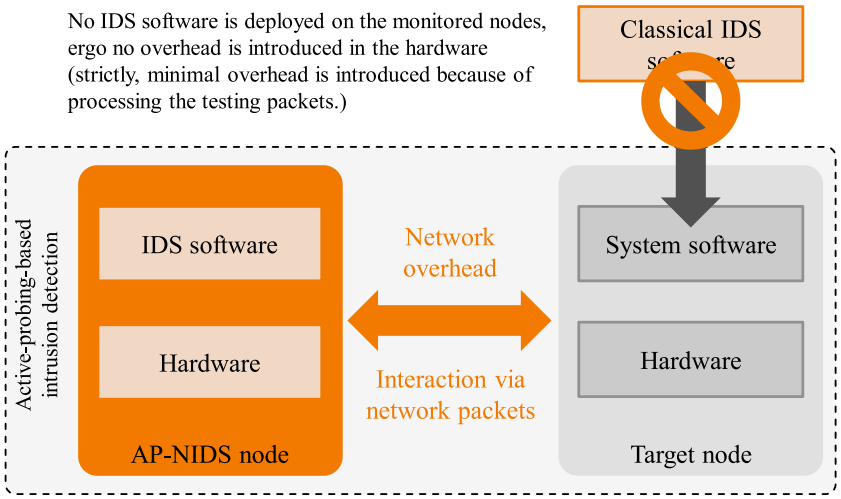


Figure 6: Conceptual depiction of the overhead in active-probing-based intrusion detection.

4.4.4 *Completeness*

"Incompleteness occurs when the intrusion detection system fails to detect an attack. This measure is much more difficult to evaluate than the others, because it is impossible to have a global knowledge about attacks or abuses of privileges" [69]. This definition is also valid for active-probing-based IDSes.

Measurement metric

The completeness can be directly evaluated by measuring the *detection rate*, i.e. the rate of attacks successfully reported by the IDS. A second metric we will employ is the *false negative rate*, which is the rate of attacks not detected as attacks.

4.4.5 *Fault tolerance*

"An intrusion detection system should itself be resistant to attacks, particularly denial of service, and should be designed with this goal in mind. This is particularly important because most intrusion detection systems run on top of commercially available operating systems or hardware, which are known to be vulnerable to attacks" [69]. There are two aspects to be considered with respect to achieving fault tolerance in active-probing-based network IDSes, particularly for WMNs. On the one hand, given that the detection software is not deployed on the node being monitored but on a separate node running as "intrusion detection node", the administrator of the AP-NIDS has total control over the software and hardware of the IDS node. In this case, the IDS node can be protected against diverse threats and this is inherent to the AP-NIDS software. On the other hand, an IDS node is exposed to threats coming from third nodes, i.e. neighboring nodes, and the IDS node should be able to detect them. For example, malicious nodes can be aware of the presence of an IDS node and stop their malicious activity when they detect that the IDS node is approaching, then continue with the malicious activity when the IDS node is out of the transmission range of them. Throughout the chapters, we refer to fault tolerance when presenting the design or configuration of a particular aspect of the intrusion detection, for example, the creation of testing packets and their selection. We provide notes and show the importance of having the fault tolerance concept in mind when setting up the IDS. We will not explicitly evaluate the fault tolerance, though.

4.4.6 *Low overhead*

Traditional IDSes introduce overhead on the resources, typically CPU usage and memory consumption, of the system that they are monitoring. Active-probing-based IDSes also introduce overhead on the physical resources of the IDS nodes where they are deployed. However, if these IDS nodes are only used for intrusion detection, it is valid to consider the software and the hardware of the IDS node as one block, where the physical resources can be fully available to the IDS software. In this case, physical overhead on the IDS node loses importance. The most important overhead introduced by an AP-NIDS is the *network overhead*. We already discussed the network overhead in relation to the efficiency in Section 4.4.1. Given the fact that the active-probing technique injects packets into the network, these extra packets are considered as extra network overhead. This concept is depicted in Fig. 6.

Measurement metric

In the different chapters of this dissertation, we will measure both the overhead on the IDS node and the network overhead. For measuring the overhead on the node, we use standard metrics such as CPU consumption, memory usage, etc. For measuring the network overhead, we take the number of packets/bytes injected, absolutely and per unit time.

4.5 SELECTION OF THE TARGET NODE

The target node or system to be monitored is completely different in active-probing intrusion detection vs. traditional intrusion detection. In active probing the target node is a different node from the IDS node, one hop away from it, which is seen by the AP-NIDS as a blackbox: A blackbox that receives some input and produces some output. The AP-NIDS is free to test any of the nodes of the network it is protecting. Here, there are almost no restrictions on the quantity or type of nodes to target. For example, an AP-NIDS can be configured to always test some given nodes, but can also be configured to test all the nodes of the network, especially if the AP-NIDS is mounted on a mobile platform. The nodes of the network can be targeted sequentially or randomly, or following a user-defined rule.

Regarding the restrictions on selecting the target node, the quality of the wireless channel to a node is the most problematic. The poor signal level of a neighboring node can cause an increase in false positives returned by the AP-NIDS, for example, by interpreting the lost packets as an intentional dropping of packets. This can be easily prevented if the IDS node replicates the testing packets transmitted and also if it only analyzes those nodes that have a signal level over a certain threshold. In the following chapters we will show that this threshold can be measured and we will observe what happens if it is correctly configured.

4.5.1 *Hiding the AP-NIDS*

Of course, the selection of the target node is related to the stealthiness of the AP-NIDS that we want to achieve. For example, advanced attackers might detect the presence of a mobile IDS node if it always chooses the nodes in a same order, and if it always follows the same moving and testing patterns. Some practical tips on making the presence of the IDS as stealth as possible might include keeping the links updated exactly in the manner a normal node does, changing the MAC address periodically but with the same rate nodes coming in and out of the network. Alternatively, target nodes can be selected randomly after all nodes are checked. The transmitted testing packets should be modeled after other packets in the network. They should not violate the routing protocols employed. The goal of this dissertation is to introduce the novel concept of active probing for intrusion detection and to show that it works. An in-depth practical analysis on hiding the IDS node and best techniques is out of the scope of this dissertation and will be left as future work.

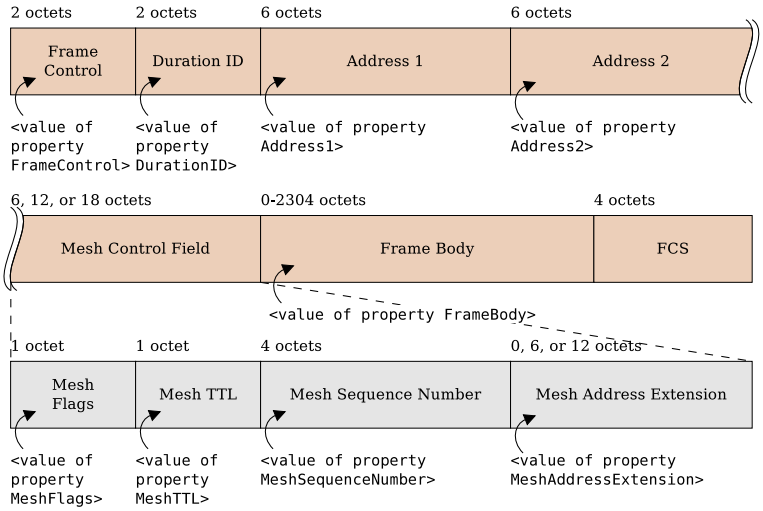


Figure 7: Layout of a testing packet for the IEEE 802.11s mesh standard. The format of the frame corresponds to the MAC protocol. Each field is filled by the IDS as required by the packet structure.

4.6 THE TESTING PACKETS

The testing packets are one of the key components of an AP-NIDS. The testing packets are responsible for creating a reaction on the target node that is analyzed afterward.

4.6.1 *Structure (composition)*

In principle, a testing packet[2] is a group of bytes injected into the network by the AP-NIDS. The structure and content of a testing packet is, technically, not conditioned. In most cases, given that the active-probing-based attack detection is performed by examining the reaction of the target node to the testing packets, these should be crafted in a way that, in case of attack, the malicious node produces (or not) a certain expected output (network traffic). For example, if the AP-NIDS wants to examine if the target node drops UDP packets with a certain port number, the testing packets it transmits are UDP with that specific port number.

Example of a testing packet

In case of an IEEE 802.11s network, the testing packets are frames whose layout complies with this standard. For example, Fig. 7 depicts the layout of a testing packet for an IEEE 802.11s network. For the implementation of the AP-NIDS that we use for the evaluation of this dissertation (details provided in a later section), we employ a modular structure: The testing packets are defined according to the protocols employed, and each field (the so-called properties in Fig. 7) of the protocol can be filled according to the detection rules. The detection rules are generated by the user of the AP-NIDS according to the attacks targeted.

4.6.2 *Testing packet selection*

There are many different testing packets that can be created for detecting different attacks. Detecting a particular attack might also need the transmission of testing packets with different morphology. For this reason, in our deployed AP-NIDS, we define the testing packets within the context of a probe (see Fig. 4). The AP-NIDS only selects which probe to run, and the

2 Note that in this dissertation we will refer to the term "testing packets" generically, even when for the concrete example of IEEE 802.11s networks, these are technically named frames instead of packets. We prefer to use the term testing packets for simplicity and homogeneity along other routing protocols.

testing packets defined in the probe are transmitted. For example, for detecting the dropping of TCP packets, we do not only transmit packets with TCP data, but a Path Request (PREQ) as well.

4.6.3 *Expected reaction*

The reaction of the target node to the transmitted testing packets is essentially the working principle of the active-probing-based intrusion detection technique and part of our innovation. We create the testing packets in a way that they originate an action on the target node that can be correlated to specific malicious behavior. For example, if we want to detect the black hole attack, sending a PREQ will result in the target node answering with a Path Response (PREP) packet claiming to have the best route to the desired destination.

4.6.4 *Loss of testing packets and other undesired effects*

The testing packets should be successfully delivered, received, and processed by the target node. From a design perspective, there are effects that are undesired, but not unexpected. We cannot predict when the loss of a testing packet will occur, but we expect this to happen under certain conditions. For example, harsh links can cause packet loss, or high network traffic can cause congestion. The study of the effects than can bring an AP-NIDS to fail in detecting attacks and their minimization is of high importance. We study these effects in Chapter 5 of this dissertation.

4.7 THE ATTACK DETECTION

The detection of attacks with active probing is conceptually based on an *action—expected reaction* philosophy. In IDS terms, it corresponds to a mixture of specification-based and knowledge-based detection model (see definitions in Section 2.3.1, page 14). If we want to detect attack x by means of active probing, we perform an action to detect it, i.e. transmitting a testing packet to the target node. The AP-NIDS knows beforehand that for a positive diagnosis of x, reaction y is expected at the target node. The reaction we refer to can be any kind traffic, or absence of traffic, produced by the target node as a consequence of x. The AP-NIDS listens to the wireless medium, gathers all the traffic produced by the target node and by any other node within its coverage area, and analyzes the traffic afterwards. The active-probing setting is vulnerable to detection errors if the active procedure fails. For example, if the testing packet sent is lost due to physical reasons, the target node will not react anymore. Part of this dissertation addresses the prevention of this kind of failure. An AP-NIDS can be designed to detect many attacks. We do not expect it to test the target node for all possible known attacks. The AP-NIDS can be able to correlate the reaction y to attacks other than x. Part of this dissertation aims at providing the AP-NIDS with a statistical model for probe selection and attack correlation.

In this dissertation we present the idea, design, and evaluation of an active-probing-based intrusion detection system. The convenience of using an active-probing technique in a practical setting will depend on many factors, such as the attacks aimed to be detected, the speed of detection, the type of network. For example, an AP-NIDS can be employed only for detecting complex attacks, with periodical analyses, while conventional IDSes can still be deployed for fast detection of other attacks.

4.8 ARCHITECTURE OF OUR IMPLEMENTED AP-NIDS: DOGOIDS

For the evaluation of this dissertation, we designed and implemented an open-source AP-NIDS. We designed an architecture, shown in Fig. 8. Our AP-NIDS is called *DogoIDS*, we implemented it in Python, and the source code and its documentation is available online

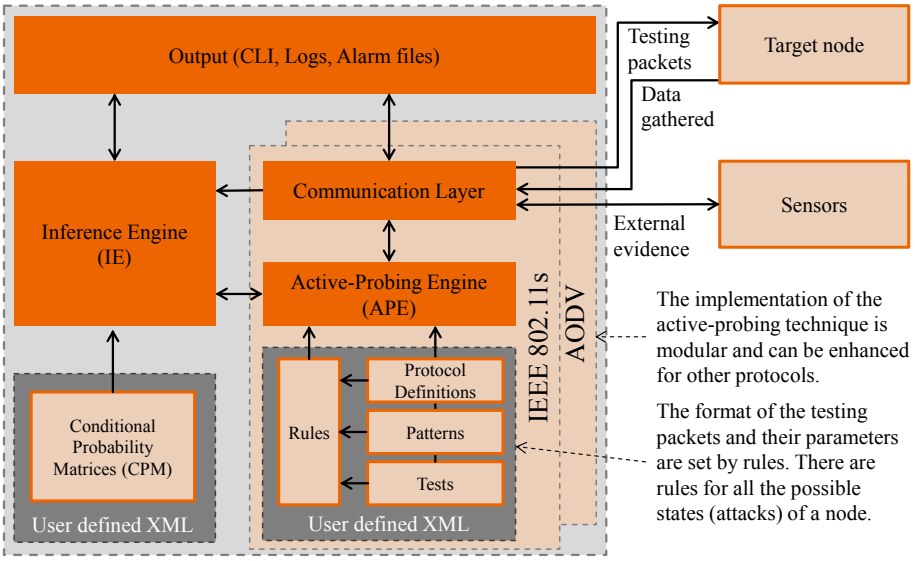


Figure 8: Overview of the architecture of DogoIDS.

at `http://sourceforge.net/projects/dogoids`. In this section, we briefly describe our architecture. We highlight that the analysis of this dissertation is general. However, for evaluating and for providing the research community with a practical AP-NIDS, we implement DogoIDS with the described architecture. Of course, the active-probing-technique is not limited to DogoIDS and can be implemented in any other system with a different architecture.

4.8.1 *The active-probing engine*

The Active-Probing Engine (APE) manages the active-probing analysis, interprets the rules, and handles the testing packets. The *rules* describe the steps involved in detecting a certain attack, including the testing packets used. The so-called *protocol definitions* define a default structure and default contents for the different protocols employed. Examples of protocol definitions include IEEE 802.11, IEEE 802.11s, Ethernet, IPv4, LLC, TCP, UDP, ARP and DNS. When DogoIDS users aim to create a rule that uses testing packets with TCP data, they can call the protocol definitions and have a template that can be modified as needed. After the process of actively transmitting testing packets and information gathering, the APE performs a set of *tests* to the data gathered. The tests are composed of internal functions and they outcome is compared to *patterns*. The aim of these tests is not to detect an attack directly, but to analyze the reaction of the target node. The results of the tests at the end of the active-probing analysis are combined in an inference engine and used to infer an attack. An example of a test is to check if the testing packets were forwarded by the target node or not. All the files needed to configure DogoIDS are XML files.

4.8.2 *The inference engine*

The Inference Engine (IE) is a temporal-selective Bayesian classifier that infers the state of a network node under test. It allows classifying if a node is misbehaving or not, based on the outcome of the probes executed by the APE. After each probe is executed, its results are injected in the IE. The Inference Engine features a recursive probe selection scheme that tells the APE which probe should be executed next. It is based on the current posterior of the Bayes classifier and a prediction step. The engine facilitates reducing the number of active probes while maximizing the insights gained by the set of probes executed. The probabilities involved are handled by a user-defined Conditional Probability Table (CPT). As this is a key

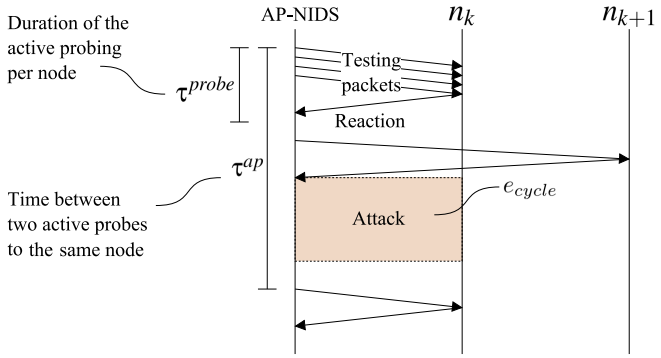


Figure 9: Conceptual description of the cycle detection error.

component of our active-probing-based network intrusion detection, we study it in detail in Chapter 6 of this dissertation.

4.8.3 *Communication layer*

The Communication Layer of DogoIDS handles the wireless communication between the AP-NIDS and the target node. It also enables gathering the network traffic present during or after the transmission of testing packets.

4.8.4 *Extensions: sensors*

The IE of the AP-NIDS can also be fed with external information. Note that this is not necessary, but a possible feature. The information can include, for example, different measurements provided from sensors deployed throughout the network. It can also be detection information from other IDS nodes. We will discuss this aspect in Chapter 8.

4.8.5 *Output*

DogoIDS provides the user with several sources of data. The active-probing process is detailed in the console as well as in log files. When an attack is detected, alarm files are generated and stored. What to do with the alarms is out of the scope of an intrusion detection system, and out of the scope of this dissertation. The alarms can be, for example, taken by an intrusion prevention system which takes a reactive action against the intruder. Alternatively, an administrator could be notified.

4.9 ISSUES OF THE ACTIVE-PROBING INTRUSION DETECTION

Up to this point, we have presented our idea and its benefits, as a solution to the problems of intrusion detection in wireless multihop networks. In this section, we also aim at identifying some of the notable issues of our proposed active-probing technique for intrusion detection.

4.9.1 *Cycle detection error*

One of the problems associated with actively probing the nodes is that an attack can be executed from a node currently not covered by the AP-NIDS. If we consider the mobility of the IDS node, in a network with K number of nodes, after the node n_k is actively probed, it is not going to be probed again until the AP-NIDS has reached its area again and has already tested the other nodes. Of course, the AP-NIDS can be configured to randomly test nodes and avoid being predicted by attackers. If the node n_k has just been actively probed, and

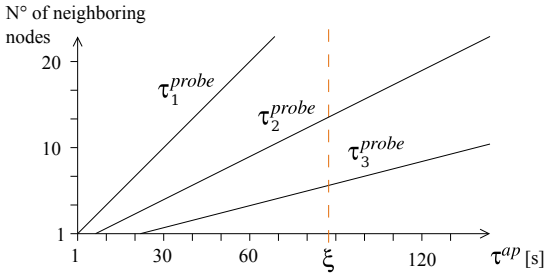


Figure 10: Conceptual plot of the relation between the duration of the active probes and the number of neighboring nodes.

it immediately launches an attack with a duration shorter than τ^{ap}, this attack will not be detected by the AP-NIDS. We call the error in the detection introduced by this situation *cycle detection error* (e_{cycle}). We denote with τ^{ap} the time between two active probes addressed to a same node. This concept is depicted in Fig. 9. Assuming a uniform duration of the active probing per node τ^{probe} for all nodes, the $\tau^{ap}_{n_k}$ for a node n_k is defined by:

$$\tau^{ap}_{n_k} = \tau^{nc}_{n_k} + (K-1)\tau^{probe} \tag{1}$$

where $\tau^{nc}_{n_k}$ is the uncovered time for the node n_k and we will determine its calculation later in (2).

Countermeasure

Eliminating this error would require letting $\tau^{ap} \to 0$. This would result in continuous active probing, which in turn would introduce a considerable overhead on the node and on the network, and could even be seen as a flooding attack caused by the AP-NIDS. In practice, we cannot fully eliminate this error but we can define an acceptable threshold ξ for τ^{ap}. This threshold denotes which is the maximum time allowed for an attack to be launched on the network without being detected. The threshold will depend on the time needed for each active probing and the average number of neighboring nodes that the AP-NIDS has at a given time: $\xi \leqslant \tau^{probe}N$, where N is the average number of neighboring nodes. Based on the density of nodes of the network, the network administrator can determine how many neighboring nodes the AP-NIDS should probe. If the active-probing-based network intrusion detection system cannot cover all the neighboring nodes without violating the threshold ξ, more IDS nodes have to be deployed. The randomization of the nodes to be tested is also a valid countermeasure.

4.9.2 *Attacks that are launched after a long period of communication*

The duration τ^{probe} of the active probing strongly depends on the type of attack to detect which determines how many testing packets to transmit and how much time is needed to capture data. If the attacker launches the attack after a certain amount of data are sent to the node, the AP-NIDS would need to transmit these data and the τ^{probe} would be necessarily longer. This would provoke that the condition of the threshold ξ might not be satisfied. Figure 10 conceptually depicts this problem. The figure shows three plots for different values of the τ^{probe}. They can be interpreted as the duration of the active probing needed for three different kind of attacks. The line for τ^{probe}_1 corresponds to attacks that are quickly detected, and therefore the IDS can probe a larger number of neighboring node without exceeding the threshold ξ. At the other extreme, the line for τ^{probe}_3 could be the case of an attack that requires the transmission of a given amount of data before being executed. Hence, the AP-NIDS can only probe a lower number of neighboring nodes without exceeding ξ.

Countermeasure

A possible countermeasure to this problem would be the deployment of two (or more) IDS nodes. One AP-NIDS is set up to probe a larger number of neighboring nodes as possible, with shorter τ^{probe}. The other AP-NIDS is set up to transmit a large amount of data to each neighboring node. This last IDS node needs more time to probe the nodes (larger τ^{ap}), but it can still detect this particular attack behavior. Clearly, in this case the network overhead introduced by the AP-NIDS is high. However, we can dramatically reduce it if we set up the second IDS node not to transmit testing packets with data but to transmit real packets of the network (the AP-NIDS is still a network node and it should forward packets when it is needed). If necessary, after the AP-NIDS forwards a given number of real data, it can send the testing packets with data crafted to detect the attacks. The network has a faster AP-NIDS aimed to detect a wide range of attacks, and in addition an AP-NIDS which is slower in detection time but efficient for some particular attacks. The administrator should consider the trade-off between detection time and network overhead introduced by the IDS nodes.

The active-probing technique requires a different approach for the setup of the intrusion detection systems. We have not covered (and cannot) all the possible issues related to the active probing. However, the problems mentioned are examples of how these potential drawbacks can be easily avoided if the AP-NIDS is correctly configured.

4.9.3 *Nodes that are not covered in a large area*

Given a network which has K nodes, the AP-NIDS will typically not cover the entire area at once but only a portion of its extension, containing a fraction of the M nodes. Assuming the AP-NIDS to be mobile, it moves within the area of the network, and at a given time it covers a different fraction of nodes. If an attack takes place in one of the nodes that is momentarily not within the coverage area of the AP-NIDS, it cannot be detected. Assuming a uniform distribution of nodes, the area covers a region $S_{network}$ which is composed of the area S_i of each node. The AP-NIDS has a coverage area $S_{IDS} = S_i$. The time that a node n_k is not covered by the AP-NIDS depends on the extension that the AP-NIDS has to cover and its area coverage rate (area covered per unit time) λ. This time is denoted by $\tau^{nc}_{n_k}$ and given by:

$$\tau^{nc}_{n_k} = \frac{S_{network} - S_i}{\lambda} \tag{2}$$

Countermeasure

The duration of $\tau^{nc}_{n_k}$ influences the duration of the active probing to the same node (τ^{ap}). Hence, the threshold ξ is also valid for this case. If the area is large and contains a high number of deployed nodes, it might provoke that one AP-NIDS cannot satisfy $\tau^{ap} \leqslant \xi$. The solution is simply to deploy more IDS nodes, which reduces the area that each IDS node needs to cover.

4.9.4 *Moving the AP-NIDS in a static network*

Another problem regarding the mobility of the IDS is the problem of deploying a mobile AP-NIDS on a static (or mostly static) network (e.g., in an industrial control system setting). In this case, the attacker can advertise the presence of a mobile node and identify which traffic corresponds to the active probes sent by the AP-NIDS.

Countermeasure

This issue can be easily solved if for this particular scenario the IDS nodes deployed do not move and they are scattered in order to cover the entire area of the network. This solution clearly breaks with our idea of letting the AP-NIDS move. However, we show that the

mobility of the AP-NIDS is beneficial for an active-probing-based IDS, and if rather particular deployments exist where the mobility is not possible, the active-probing technique can still be used with the higher cost of deploying more IDS nodes.

4.10 SUMMARY

In this chapter, we introduced our active-probing technique for intrusion detection in wireless multihop networks. The concept and design has been established. We have shown that our solution for intrusion detection can be to launch from a single node dedicated to intrusion detection. This concept frees us from the need to deploy IDS sensors on every node of the network, eliminating the burden of the overhead on them and saving deployment troubles. We showed that the key point of our intrusion detection is that it actively transmits testing packets to its neighboring nodes. In the next chapter, we will cover the testing packet transmission and correct parametrization of the AP-NIDS for reducing detection time and detection errors. We have also introduced the architecture of the proof-of-concept AP-NIDS, DogoIDS, that we employ for the evaluation of this dissertation. The architecture of DogoIDS helps to understand how the experiments of the rest of the dissertation have been produced. In addition, the active-probing technique we propose is theoretically general, though, DogoIDS and its architecture materialize the active-probing intrusion detection and present us with practical challenges that need to be tackled. For example, in the following chapters we show that a statistical method for probe selection and attack inference helps in reducing the amount of probes executed.

5 ON THE PARAMETRIZATION OF AN AP-NIDS

> The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.
>
> William Bragg

In this chapter, we study the interaction between an AP-NIDS with the target node. We identify critical parameters and we adjust them empirically.

5.1 INTRODUCTION

Given that there must be an interaction between an AP-NIDS and its target node through testing packets, it is important to ensure that the testing packets are successfully delivered, received, and processed by the target node. The packets generated by the target node after active probing should also be received and processed by the AP-NIDS. The correct operation of the active probing mechanism has to be ensured even under adverse channel conditions and in case of congested nodes, which could easily lead to false detections.

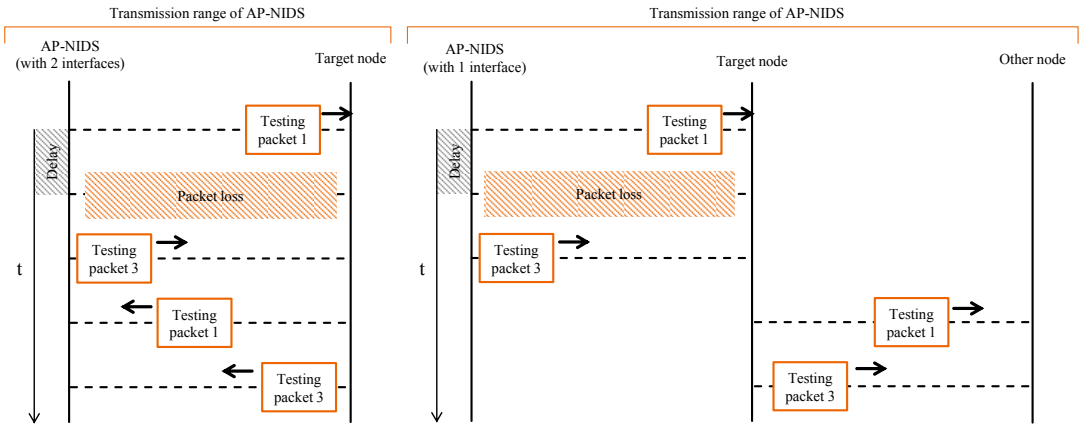
The general goal of this chapter is to understand the interrelationship of an active-probing-based IDS with real WMNs. The objectives of this chapter in particular are, first, to understand which parts of the interaction AP-NIDS–target node are critical. Second, the chapter aims to identify the causes that can provoke an undesirable interaction AP-NIDS–target node. Finally, we analyze the parameters of an AP-NIDS that can be properly set in order to avoid an undesirable interaction [87].

The testing packets use the same network (same protocols/functions) as the nodes to test and therefore the AP-NIDS has to ensure that the network works, or at least the underlying networking. It is obvious that if the AP-NIDS want to test attacks that run on the network layer (i.e. dropping of UDP packets), the physical and data link layers of the target node should work properly. The parameters that we discuss in this chapter aim to guarantee that the underlying layers work correctly.

This chapter is organized as follows. In Section 5.2 we start by briefly explaining the interaction between the IDS node and the target node. In Section 5.3 we identify the important parameters that have to be adjusted in our system. We explain why these parameters are important and how we adjust them. In Section 5.4 we describe the experiments that we perform for adjusting the different parameters, and later describe and discuss the results obtained in Section 5.5. Finally, Section 5.6 concludes this chapter.

5.2 OVERVIEW OF THE INTERACTION AP-NIDS–TARGET NODE

We briefly describe the interaction between the AP-NIDS and the target node. There is no fixed rule about how the testing packets should be composed and their behavior, but we provide two generic approaches. In the example of detecting the dropping of packets attack type, the procedure is to transmit testing packets to the target node and observe if they are dropped. The principle is easy. In practice, the testing packets need to be addressed to a third node, in other words, the final destination (AP-NIDS transmits them to the target node and the target node forwards them to the final destination). The third node can be, for example, another node of the network which is a neighbor of the target node. In a different setting, if we want to be sure that the third node is a neighbor of the target node, we can build the AP-NIDS with two network interfaces. One is then used to transmit the testing packets and



(a) Testing packet transmission when the AP-NIDS has 2 interfaces.

(b) Testing packet transmission when the AP-NIDS has 1 interface.

Figure 11: Possible configurations for the testing packet transmission.

the second one takes the role of the final destination. In more complex configurations, a third node can be another AP-NIDS node, but this is not a requirement. Both scenarios are depicted in Fig. 11. We identify two important issues of the packet transmission that can have a critical influence on the active probing intrusion detection: the physical packet loss and the packet loss due to congestion (or other reasons). In order to minimize them as much as possible, we identify three parameters that can be set: the threshold for the signal level to the target node, the repetition/replication of testing packets, and the time interval between repetitions of testing packets. We describe them in more detail in the following.

5.3 PARAMETRIZATION OF THE ACTIVE PROBING

In this section, we identify and describe the most important parameters to adjust in active probing.

5.3.1 *Threshold for the signal level*

On the physical layer, we employ the wireless medium for communicating with the target nodes, and the wireless medium is sensitive to physical effects such as noise, fading, etc. The testing packets that the AP-NIDS sends in order to test the working state of the target nodes should always arrive to them. If they are lost because of a harsh channel, the AP-NIDS would misinterpret that loss as an intentional packet dropping. For this reason, we consider that the AP-NIDS should perform active probing only if the link between the AP-NIDS and the target node is good enough. Selecting good links is not a limitation. WMNs are dynamic and their participating nodes move in and out of the transmission range of the intrusion detection system. For this reason, it is not a problem to focus on those nodes that are close enough. In addition, if the AP-NIDS is mounted on a mobile platform, it will cover the complete network at some point in time, making it easier to get close enough to the nodes and get good links.

We recall that our focus is not on the physical layer, and we do not cover attacks that target the physical layer. The goal of measuring the quality of the link between the AP-NIDS node and the target node is only for an effort to guarantee that the packet delivery succeeds, and test the upper layers. We will set a threshold for the signal level by performing attack detection against nodes with different signal levels.

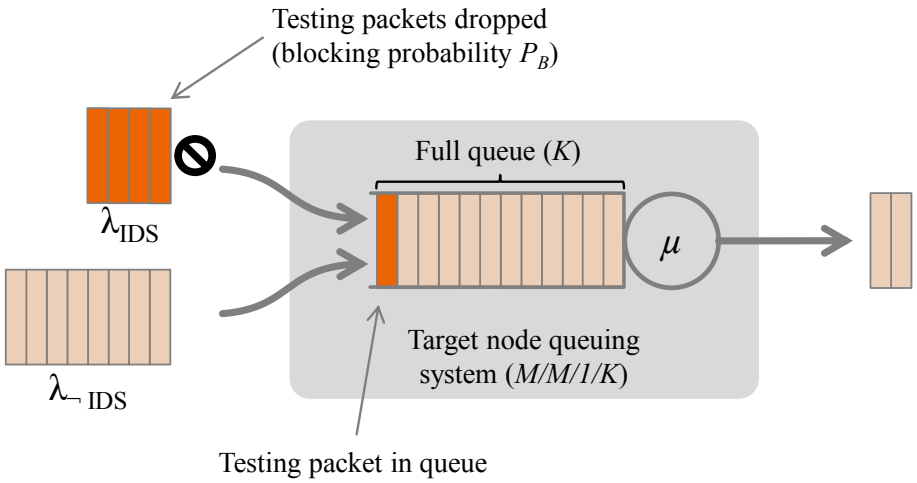


Figure 12: Representation of the queueing system of a target node and the full queue problem.

5.3.2 *Repetitions of testing packets*

In theory, a node in a wireless multihop network is a queuing system with a finite buffer. The AP-NIDS cannot know how the queuing system of the target node internally works, since the target node is a black box which is measured from the outside. For this reason, and for the sake of simplicity, we model a network node as a $M/M/1/K$ queuing system as depicted in Fig. 12 [88]. We assume a FIFO queuing. K is the capacity of the target node, i.e. the maximum number of packets in the target node (queue plus processing) including the one under service.

Let λ be the traffic arrival rate, which represents the incoming packets into the target node from all independent sources, including the testing packets (λ_{IDS}), in other words $\lambda = \lambda_{IDS} + \lambda_{\neg IDS}$. All these random variables are independent of each other. μ is the processing or service rate, i.e. packets per unit time that have been processed and have left the system. $N(t)$ is the number of packets in the queue at the time t, and W the average time a packet spends in the queuing system (including time in the queue and processing time). $\rho = \lambda/\mu$ represents the fraction of time the system is working (processing packets). Given that the buffer is finite with capacity K, the equilibrium condition is given when:

$$\rho = \frac{\lambda}{\mu} < 1, \tag{3}$$

in other words, the processing rate should be faster than the packet incoming rate. In a practical WMN this condition does not always occur. A congested node is not desirable, however when this is the case and the queue of a node is full, incoming packets are discarded until the queue is free again. Here resides the problem regarding active probing. If at the moment of transmitting a testing packet, the queue of the target node is full, the testing packet will be dropped by the target node. This is a very unfavorable situation since it might lead the AP-NIDS to believe that the packet drop is due to malicious activity instead of a full queue, and this will, consequently, increase the false positive rate. The probability that an arriving packet (any packet) find the target node full (K packets in it) is the well-known blocking probability P_B [88], denoted as:

$$P_B(K, \rho) = \frac{\rho^K(1-\rho)}{1-\rho^{K+1}} \tag{4}$$

However, calculating the blocking probability is not a practical solution as all the internal parameters (queuing model, K, μ) of the target node are unknown. The only adjustable parameter from the AP-NIDS is the number and rate of transmitted testing packets. Hence, a

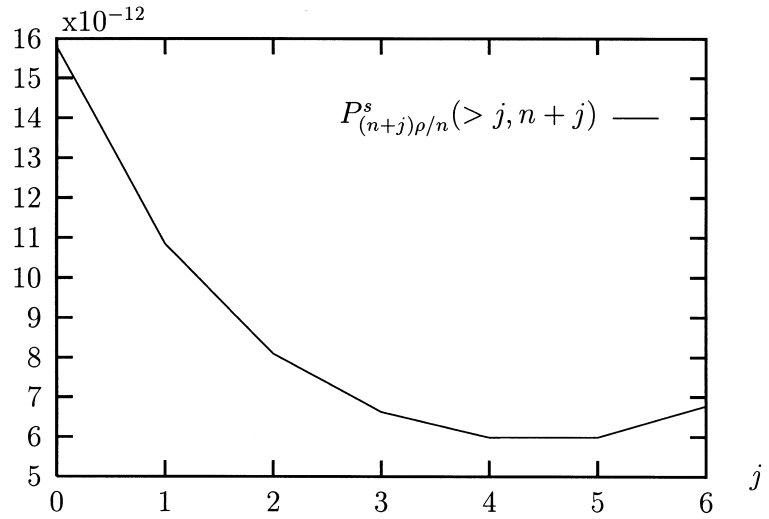


Figure 13: Loss probability as a function of j redundant packets (from [90], p. 506, Fig. 4). In this example, adding four redundant packets decreases the loss probability more than adding up to three. Exceeding four causes the loss probability to increase.

practical solution to reduce the packet loss, or at least, to guarantee that some of the testing packets are processed by the target node is redundancy of testing packets. The effects of adding redundancy to a M/M/1/K queue in order to reduce loss probabilities has already been analytically covered by Altman and Jean-Marie in [89] and Ait-Hellal et al. in [90]. In [90] the authors formally demonstrate that adding j redundant packets to a group of n consecutive packets transmitted from a given source S decreases the loss probability. They define $P^S_\rho(> j, n+j)$ as the probability of no more than j losses in a group of $n+j$ packets. If at least n packets of the group arrive to the destination, then the packet can be reconstructed (no loss). We will not cover the formal demonstration of this analysis since it can be directly found in [90]. However, it is in our interest to cite the conclusions they obtained, since they are applicable to our work. First, they demonstrated that adding redundancy does decrease the loss probability. And secondly, they showed that adding redundancy does not linearly decrease the loss probability, but it increases it if the number of redundant packets is greater than a certain value. Figure 13 demonstrates this phenomenon.

The goal of our measurements is to experimentally show that the increasing repetition of testing packets up to some point helps reduce the packet loss, and therefore, improve the detection quality.

5.3.3 *Interval between repetitions*

If the queues of the neighboring nodes are full, flooding them with repetitions of testing packets increases the congestion and network overhead. However, choosing long time intervals between repetitions delays the overall detection time per node. We study the trade-offs of increasing and decreasing this interval, and we present the results in Section 5.5.3.

5.4 EXPERIMENTS

In this section, we describe the experiments performed for this chapter. We used an indoor wireless mesh testbed composed of 16 fixed nodes and 2 netbooks running the AP-NIDS. The configuration of two interfaces for an AP-NIDS was used (refer to Section 5.2). We performed the experiments with varying background traffic. We used transmit UDP traffic between the target node and a neighboring node with the following values: 0 Mbit/s (0%), 6.25 Mbit/s (25%), 12.5 Mbit/s (50%), 18.75 Mbit/s (75%), 25 Mbit/s (100%). The last value represents

Table 3: Kernel module *ieee80211* debugging information

Date	dropped_frames_congestion	dropped_frames_no_route	dropped_frames_ttl	fwded_frames	fwded_mcast	fwded_unicast	estab_plinks	IDS_rx_dropped	IDS_rx_bytes	IDS_rx_packets
2013-11-11 19:13:44.636	0	22	0	2955	703	2252	9	1724	1043144	22560
2013-11-11 19:13:56.004	0	27	0	2961	705	2256	9	1733	1045186	22605

the maximum achievable throughput that we obtained in our network (100%). We used UDP instead of TCP to achieve a higher throughput. The technical details and complete description of the testbed is detailed in Section A.2 of the Appendix on page 108.

5.4.1 *Threshold for the signal level*

In this experiment, we want to identify the minimum signal level of the target node that guarantees minimal packet loss. We directly measure the Received Signal Strength Indicator (RSSI) of the target node when performing active probing. We show and discuss the results in Section 5.5.1.

5.4.2 *Repetition of testing packets*

In our particular experiments we work with the IEEE 802.11s mesh standard. We set DogoIDS to transmit a PREQ before sending the testing packets. The target node receives the PREQ and builds the path to the final destination by forwarding the PREQ to its neighbors and waiting for a PREP. We present the results of this analysis in Section 5.5.2.

Inspecting the debugging information of the kernel module `ieee80211` (see Table 3) we note that for the time DogoIDS reported an attack (a false positive in this case), the counter `dropping_frames_ no_route` was increased. The counter did not increase when DogoIDS did not report an attack, leading us to conclude that the target node received the testing packets, but did not find a path to their destination. The reason is that the PREQ sent at the beginning of the analysis was not received by the target node, and therefore a route to the destination node was not created. The peer links are stable and never changed during the complete experiment (this is easy to determine by examining variations in the `estab_plinks` counter over time.)

A solution to the problem of loss of control packets is to replicate the control packets. We examined two alternatives. The first case was to transmit a control packet, in our case a PREQ, per testing packet. The second and better alternative was to replicate the PREQ before the transmission of the testing packets, for example, two times. The latter case is better because we did not transmit too many control packets. We have to experimentally determine which alternative is better, or in other words, if the second alternative is sufficient.

We performed our experiments with three different configurations: 1 PREQ before the transmission of testing packets (1 PREQ for n repetitions); 1 PREQ per repetition testing packet (n PREQ for n repetitions); and 2 PREQs before the transmission of testing packets (2 PREQs for n repetitions). We do not lose generality with our configuration: The repetition of testing packets is inherent to the employed routing protocol, as well the repetition of

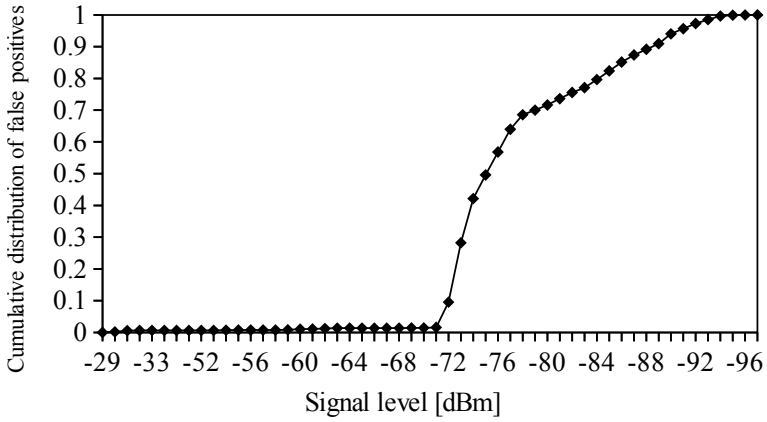


Figure 14: Study of the effects of the received signal level.

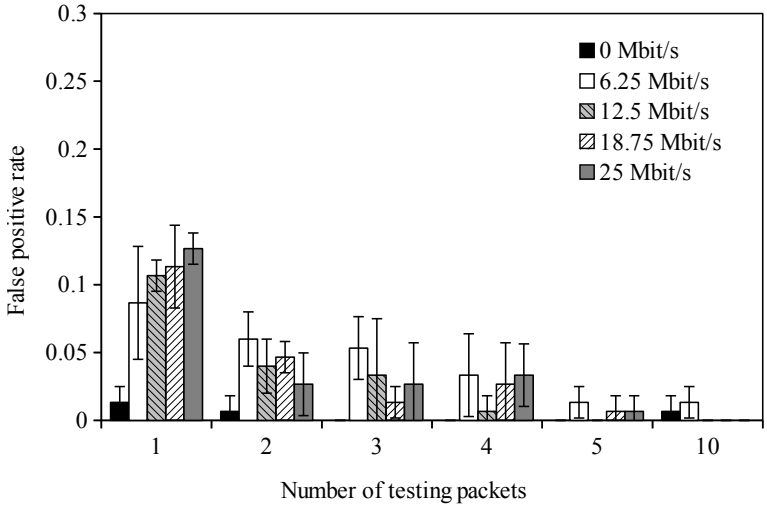


Figure 15: False positive rate for different repetitions of testing packets with 1 PREQ per analysis transmitted.

control packets. The control packets per se are, of course, protocol-specific. We performed our experiments for different background traffic levels.

5.5 RESULTS

5.5.1 *Threshold of the signal level*

Figure 14 shows the distribution of false positives for different values of the signal level. Nodes with a signal level below ≈ -70 dBm show a significant increase in false positives. Therefore, we do not recommend analyzing nodes under -70 dBm. Of course, this will depend on the device and deployment. It is critical, though, that an AP-NIDS is calibrated with respect to basic wireless parameters.

5.5.2 *Repetition of testing packets*

Figures 15, 16, and 17 show the false positive rate when transmitting 1 PREQ at the beginning of each analysis, 1 PREQ per testing packet transmitted, and 2 PREQs at the beginning of each analysis respectively. In all cases the trend is to have a lower False Positive Rate (FPR) when increasing the replication of testing packets.

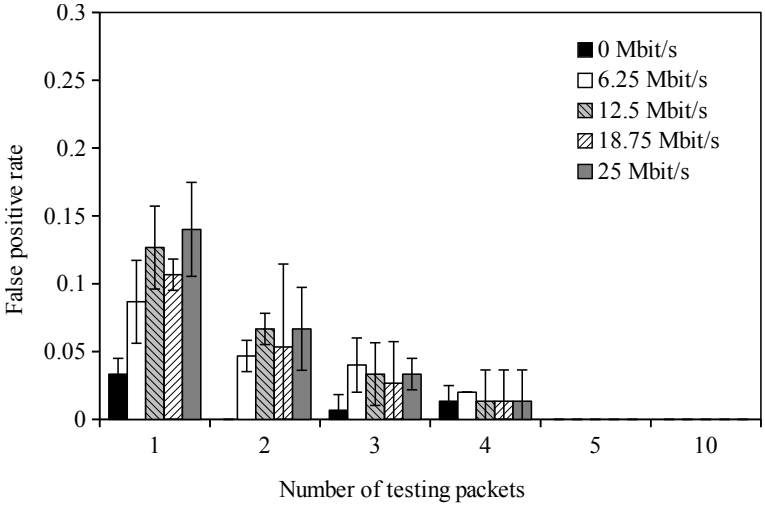


Figure 16: False positive rate for different repetitions of testing packets with 1 PREQ per testing packet transmitted.

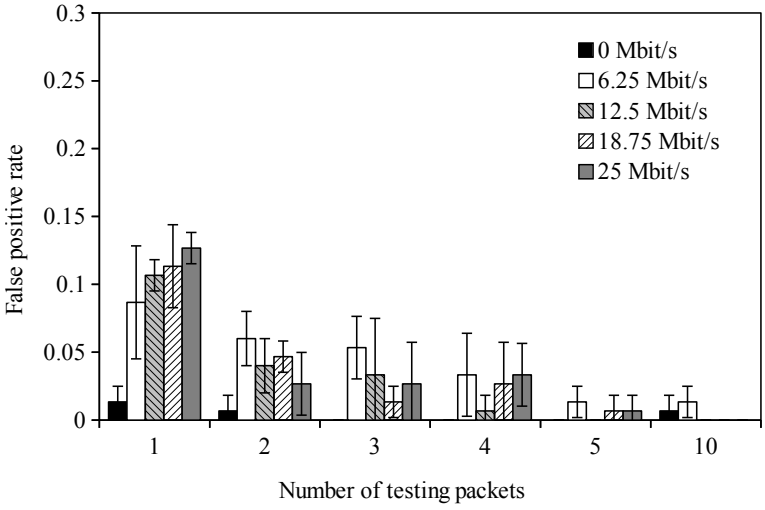
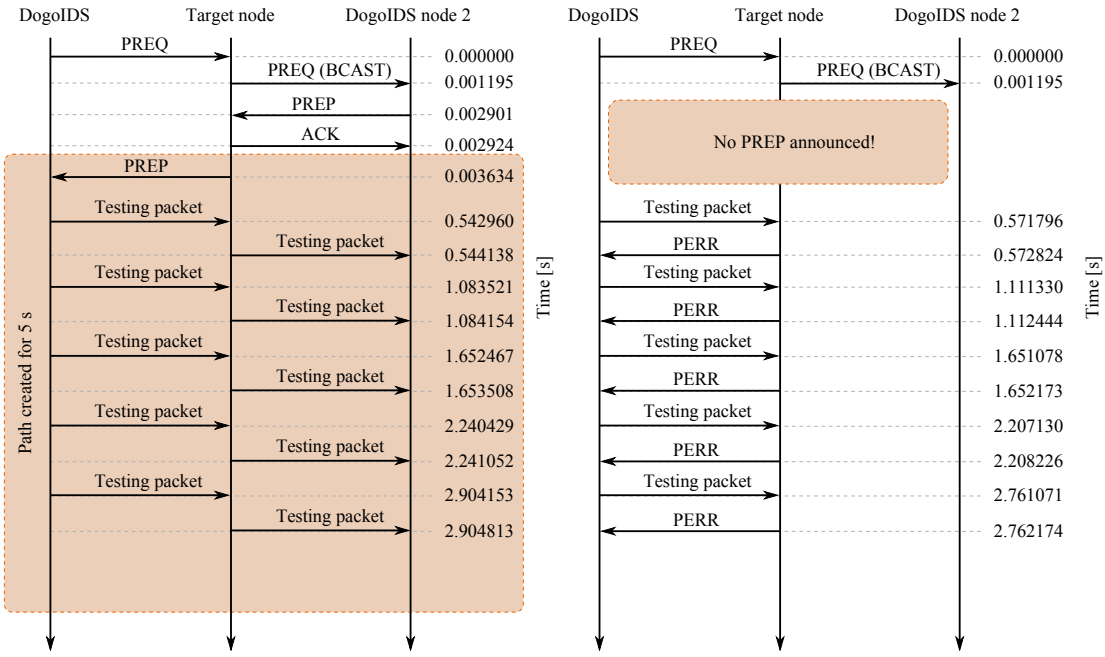


Figure 17: Duration for different repetitions of testing packets with 2 PREQs per analysis transmitted.



(a) Timeline of a successful active probing.

(b) Timeline of a failed active probing (a false positive).

Figure 18: Explanation of the false positives obtained by transmitting 1 PREQ.

Transmitting 1 PREQ before the testing packets

Transmitting 1 PREQ per analysis yielded higher false positive rates for every number of repetitions of testing packets and background traffic. We analyzed the captured `tcpdump` traces of 1250 analyses (first replication of each experiment) and observed that 88.06% of the failed probes (false positives), failed because of a routing problem. The target node generated a Path Error (PERR) message because of a missing route to the final destination. The rest of the failed probes (11.94%) failed because DogoIDS did not recognize the testing packets, even though they were correctly forwarded by the target node. The reason lies in an implementation bug while reading the captured traces. Interestingly, this phenomenon only occurred when we transmitted only one testing packet because it is unlikely that an implementation bug takes place twice or more times for the same trace if several testing packets are present. These results are shown in Fig. 19.

The reason for the missing routing paths is simple: if the PREQ is lost at the beginning, it does not matter how many repetitions of testing packets are sent afterward, they will be discarded because of a missing path. We illustrate this effect in Fig. 18. Figure 18b is a representation of the packets exchanged during one of our experiments. The experiment was set to transmit 1 PREQ and then 5 replications of testing packets. In this example, the PREQ was received by the target node and forwarded (broadcast). The destination node did not

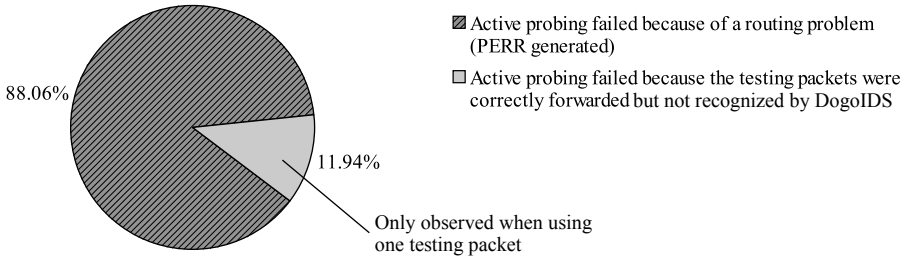


Figure 19: Cause of failure (false positives) for 1250 analyses.

answer with a PREP, and the target node did not build a path. The incoming testing packets did not have a path to the destination node, and the target node generated a PERR message per testing packet. Figure 18a shows an example where the PREQs and PREPs are present, and the testing packets are forwarded by the target node as expected.

Why a "false positive"?

For the experiments in this chapter, we set DogoIDS to test the target node against the dropping of UDP packets. The test is simple: DogoIDS transmits a set of testing packets containing a UDP packet with random ports. The target node is expected to forward these UDP packets to their destination (our second interface). We want to detect a complete dropping of UDP packets. If at least one replication of testing packets is correctly forwarded, the analysis succeeds. If not, an alarm is triggered (dropping of UDP packets). The destination node is two hops away from DogoIDS, i.e. through the target node `DogoIDS` ↔ `Target Node` ↔ `Destination node`. The target node does not have a permanent route to the destination node. For this reason, the path discovery mechanism should be invoked before transmitting the testing packets. DogoIDS sends a PREQ message to the target node, asking for the best route to the destination node. The target node and the destination node exchange path messages, and DogoIDS sends the testing packets.

However, for these experiments, DogoIDS is not configured to overhear the path discovery process, and not to detect the path error messages either. For the sake of ease of implementation of our proof-of-concept, DogoIDS transmits a PREQ and the testing packets a short time afterward. If the path discovery process fails, DogoIDS does not note it, and triggers an alarm because the testing packets were not forwarded. This situation is a false positive from the IDS perspective, because DogoIDS was not configured to differentiate between dropped packets because of a legitimate attack, or dropped packets because of a failed path discovery process. To improve this situation, we have two choices: a) we let DogoIDS be able to send testing packets after receiving confirmation from the path discovery process; or at least, we let it identify the path error messages generated by the target node; b) we send replications of path request messages in the hope that at least one will succeed. We chose the second option.

Transmitting 2 PREQs before the testing packets

We configured DogoIDS to send 2 PREQs at the beginning of each analysis and we repeated the experiments. The results of the false positive rate are shown in Fig. 17. The results improved significantly. When DogoIDS transmits only one testing packet, the FPR is comparatively similar to transmitting one PREQ. When sending one PREQ and one testing packet, the alarm can be triggered because either the PREQ was lost, or the testing packet was lost. If we send two PREQs, but one testing packet, is less probable that both PREQs are lost, however, the testing packet can be lost.

Transmitting 1 PREQ per (repetition of) testing packet

In order to examine how the loss of a PREQ affects packet forwarding on the target node, we set the AP-NIDS to transmit a PREQ with and before every repetition of testing packets sent. For example, if 5 repetitions of testing packets are sent, 5 PREQs are too. The false positive rates are depicted in Fig. 16. As it was to be expected, the FPR is lower than only transmitting one or two PREQs. With 5 repetitions of testing packets, we obtained zero false positives. The analysis duration is higher, because more packets are sent from the AP-NIDS (n repetitions of testing packets + n PREQ). Despite the lower false positives of this setup, transmitting a PREQ with every repetition of testing packets has some disadvantages. First, sending so many PREQs is a behavior pattern and the AP-NIDS could be unveiled by an attacker. Secondly, it increases the network overhead, even when it is minimal. Thirdly, it increases the duration of the analysis. We conclude that the obtained precision with the simple strategy of transmitting

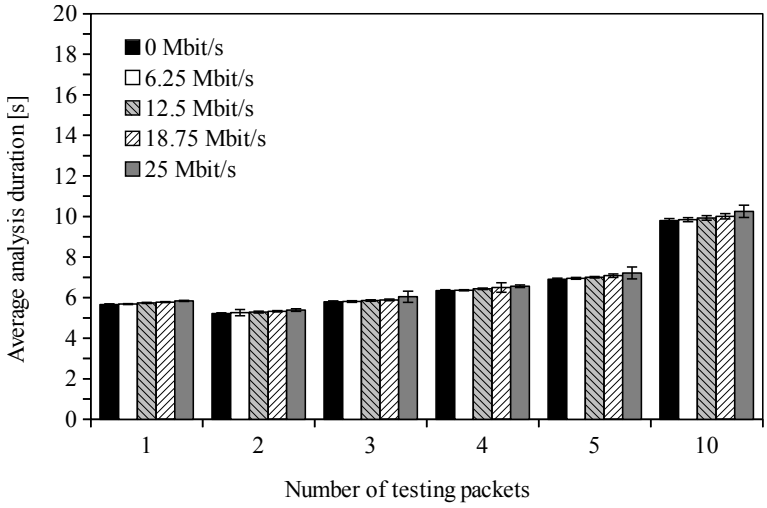


Figure 20: Duration for different repetitions of testing packets with 1 PREQ per analysis transmitted.

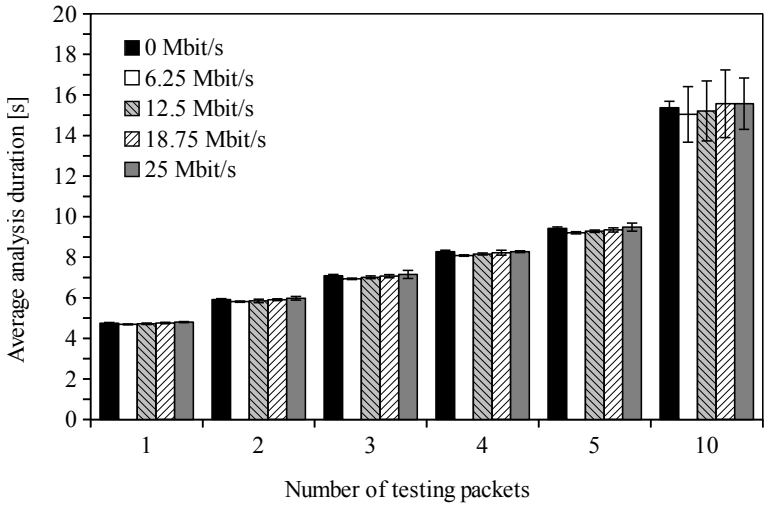


Figure 21: Duration for different repetitions of testing packets with 1 PREQ per testing packet transmitted.

2 PREQs before the testing packets is sufficient. Alternatively, the full 802.11s control message handling could be implemented to address the observed problem actively.

Effects of background traffic

Background traffic increases congestion at the target node and, consequently, increases packet loss. For example, as it can be observed in Figures 15, 16, and 17, when transmitting only one testing packet with our AP-NIDS, the false positive rate between background traffic present and no background traffic is around one order of magnitude. The amount of data captured per analysis is, of course, bigger with the presence of background traffic. We depict the amount of data captured in Figures 23, 24, and 25. Although more data needs to be processed, it does not harm the overall analysis duration. For example, the difference between 0 Mbit/s and 25 Mbit/s of background traffic for 5 testing packets (using 2 PREQs) is about 23 ms.

How many repetitions of testing packets are better?

Figure 26 is a four dimensional plot that depicts the effects of the repetitions of testing packets with different levels of background traffic (using 2 PREQs). The increasing background traffic does not significantly increase the detection time (vertical axis). However, the increasing

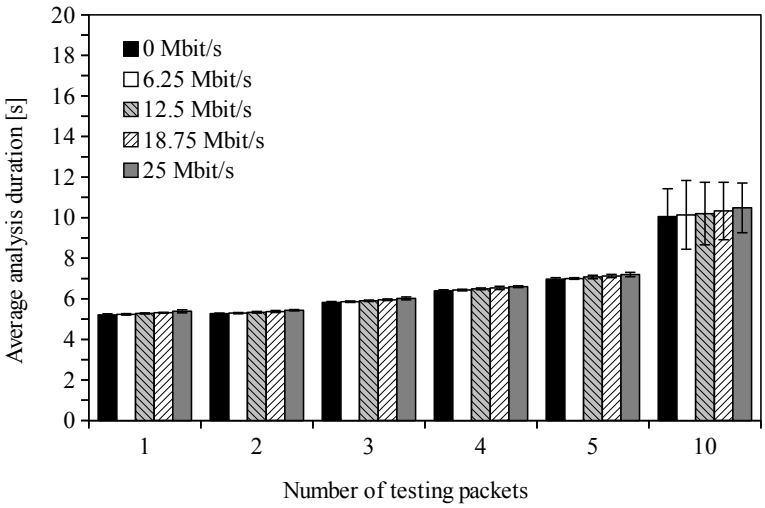


Figure 22: False positive rate for different repetitions of testing packets with 2 PREQs per analysis transmitted.

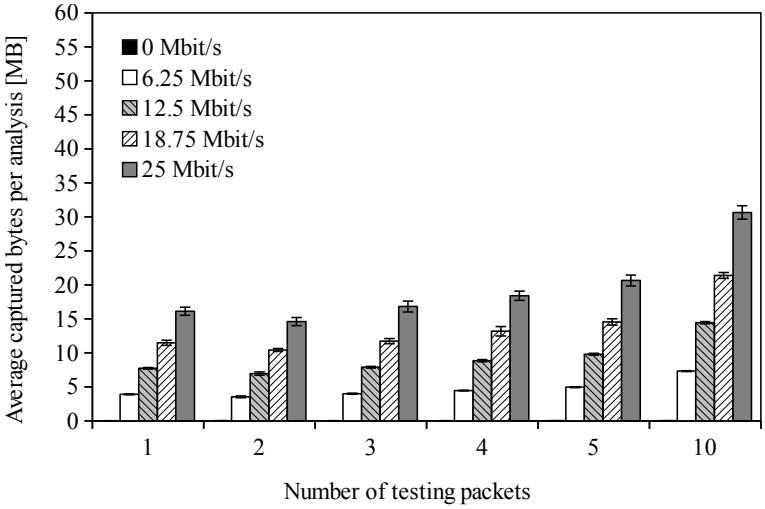


Figure 23: Average captured bytes per analysis for different repetitions of testing packets with 1 PREQ per analysis transmitted.

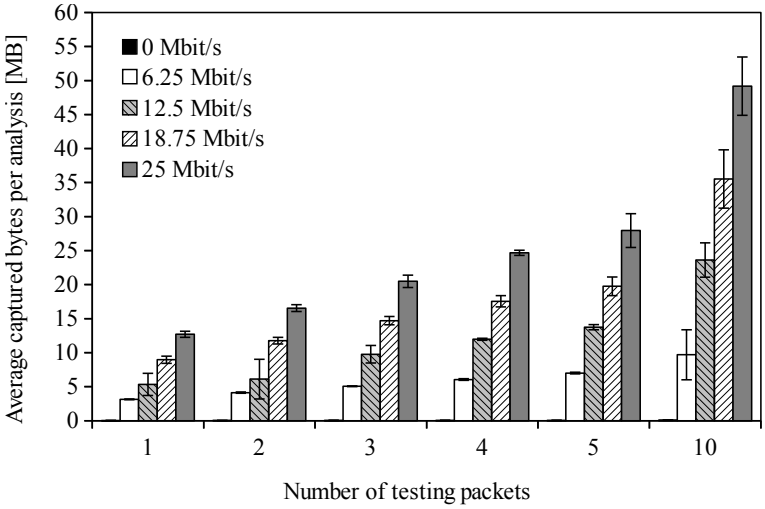


Figure 24: Average captured bytes per analysis for different repetitions of testing packets with 1 PREQ per testing packet transmitted.

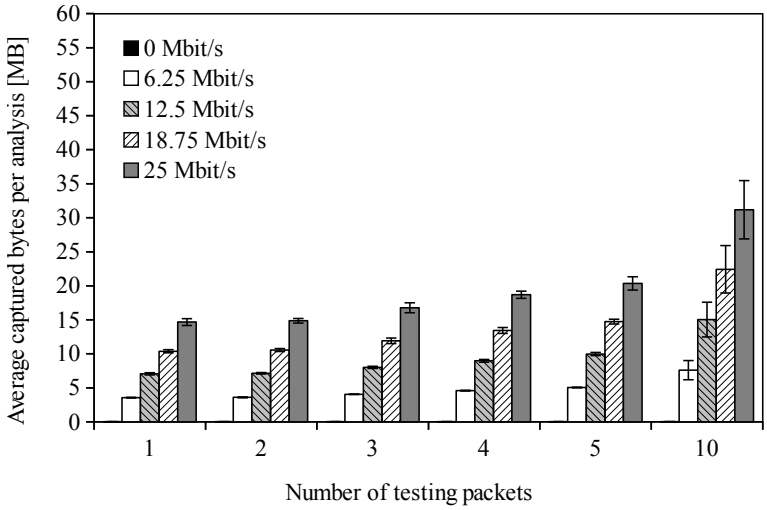


Figure 25: Average captured bytes per analysis for different repetitions of testing packets with 2 PREQs per analysis transmitted.

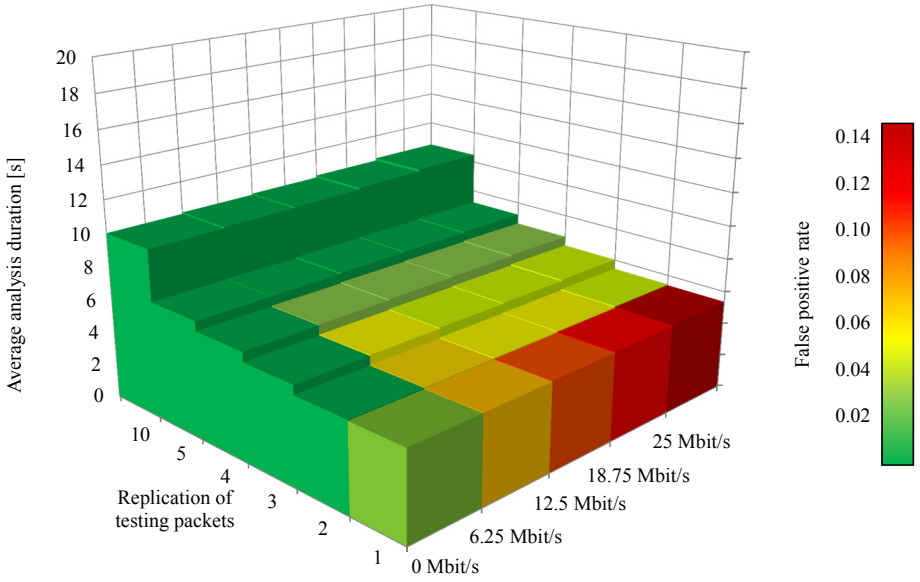


Figure 26: Average duration and false positive rate for different repetitions of testing packets when sending 2 PREQs per analysis.

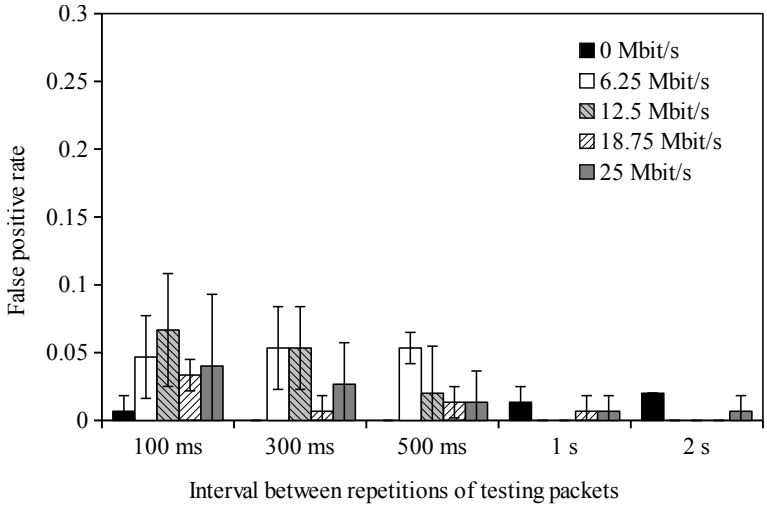


Figure 27: False positive rate for different intervals between repetitions.

background traffic drastically increases the false positives when the number of testing packets transmitted is low, namely between 1 and 3. With 4 testing packets, the false positives remain almost constant, but still higher when background traffic is present. Upwards of 5 testing packets, the false positives are low, in the 0%-1.3% zone.

The detection time and the false positives are two variables that we want to reduce as much as possible. After 5 testing packets, the false positive rate does not improve. Sending more than 5 testing packets is not favorable because it does not decrease the FPR, but increases the detection time in our setting.

Randomization of PREQ replications of testing packets

Since we are using active probing for security purposes, it is obvious that the intrusion detection system should remain stealth to the eyes of the attackers. Transmitting replications of control/testing packets with the same period can attract the attention of an attacker and unveil the presence of the IDS node. A simple method to avoid detection is to set random times between the replications of control and testing packets. Of course, these times should be between certain intervals. These intervals can be obtained from our study later in Section 5.5.3. The evaluation of the randomization of PREQs is out of the scope of this dissertation and we leave it for future research.

5.5.3 *Interval between testing packets*

Figure 27 depicts the FPR as a function of the different repetition of testing packets transmitted for intervals between repetitions of 100 ms, 300 ms, 500 ms, 1 s, and 2 s. When using low intervals between repetitions (e.g., 100 ms, 500 ms), we obtain higher false positives than when using longer values. The reason is the incapacity of the node of processing a congested queue. The arrival rate of the testing packets is simply higher than the processing rate of the queue and they are discarded. We can observe that when the target node was not loaded (0 Mbit/s), the FPR was still low for shorter intervals between repetitions. The first conclusion is that setting different values for the interval between testing packets presents a minor difference in the FPR up to a certain threshold, for example, above 500 ms.

In Fig. 28 we depict the average duration of a complete active probing cycle for the different intervals of repetitions. The average duration, of course, increases with the repetitions of testing packets. For this reason, setting values too high for the interval between repetitions makes the active probing process slower. A possible setting for an AP-NIDS in production could be to use random values for the interval between repetitions, between t_{min} and t_{max}.

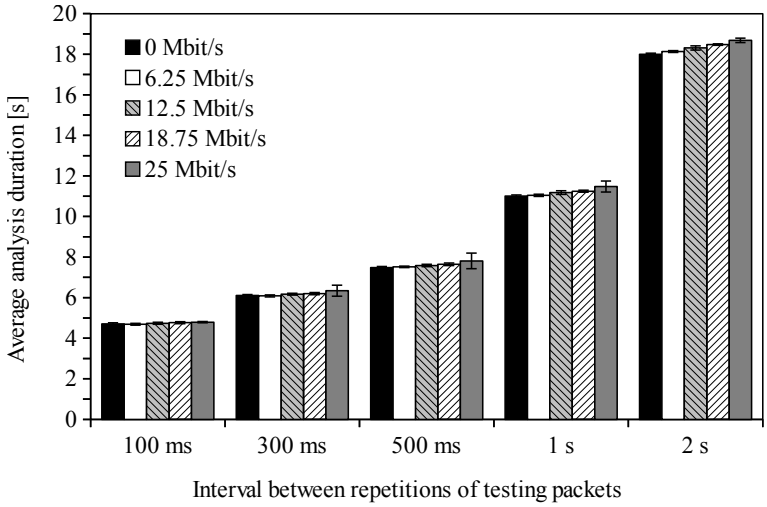


Figure 28: Duration for different configurations of interval between repetitions.

For example, if the security policies of the network cannot tolerate an FPR bigger than FPR_{max}, we can set the t_{min} as the lower interval between repetitions that does not yield an FPR higher than FPR_{max}.

5.5.4 *Overhead*

In the world of active-probing-based network intrusion detection, we distinguish two kinds of overhead. On the one hand, there is the overhead that the IDS node has for running the AP-NIDS software. This software demands CPU and memory usage. As the AP-NIDS works as an individual dedicated IDS node, we are free to deploy the AP-NIDS software on a node with enough resources. For this reason, we minimize the importance of this overhead. Of course, the hardware requirements should be of "normal" proportions in order to be cost-effective in real deployments. On the other hand, network overhead is produced by the injection of testing packets. This overhead is inherent to active-probing-based intrusion detection systems.

Overhead on the IDS node

In Table 4 we depict the average values of CPU usage and memory load when performing the experiments with 2 PREQs per testing packet. The CPU usage is not badly affected by the background traffic, even though the AP-NIDS needs to handle bigger pieces of traffic captured. The memory consumption was low, around 2% in all cases. DogoIDS, the proof-of-concept that we built for the evaluation of this dissertation, hardly put a load on the hardware of the IDS node.

Network overhead

The network overhead is the extra load that is present in the network traffic because of the use of active probing. In order to depict this extra load, we show in Fig. 29 the relation between total number of packets and testing packets present in the capture files of DogoIDS. Each capture file corresponds to one single analysis, and they contain all the traffic present by the time of the analysis (including control packets, probes, acknowledgments, etc.). Figure 29 shows the number of packets for five different analyses, each one for a different value of background traffic, and when the AP-NIDS was set to transmit 5 testing packets. When there is no extra background traffic present, the testing packets (plus 2 PREQs) represents 16.22% of the total packets. The number of packets that the AP-NIDS sends does not depend on the background traffic and remain constant. When the background traffic is of 6.25 Mbit/s, the

Table 4: Average CPU usage and average memory load for DogoIDS for 2 PREQs per analysis

	CPU usage				
	0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
1	74.99% ± 2.91	74.15% ± 4.13	70.65% ± 6.94	70.20% ± 7.84	69.43% ± 7.83
2	75.20% ± 3.74	73.53% ± 3.17	69.84% ± 8.23	70.54% ± 7.39	70.81% ± 7.56
3	78.13% ± 7.45	75.36% ± 5.23	73.32% ± 7.47	73.53% ± 7.23	72.63% ± 8.14
4	80.02% ± 5.58	77.11% ± 4.96	74.50% ± 7.95	74.92% ± 7.91	74.33% ± 8.12
5	82.19% ± 5.97	77.64% ± 6.67	77.02% ± 7.07	76.18% ± 8.20	75.77% ± 7.62
10	86.80% ± 7.45	83.27% ± 7.12	82.02% ± 8.75	83.57% ± 6.61	82.81% ± 6.61
	Memory load				
	0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
1	2% ± 0.07	2% ± 0.05	1.99% ± 0.08	1.99% ± 0.11	1.99% ± 0.07
2	1.99% ± 0.08	2% ± 0.06	1.99% ± 0.09	1.98% ± 0.10	1.97% ± 0.12
3	2% ± 0.04	2% ± 0.05	1.99% ± 0.11	1.98% ± 0.11	1.95% ± 0.13
4	2% ± 0.04	2% ± 0.06	1.99% ± 0.08	1.96% ± 0.12	1.93% ± 0.14
5	2% ± 0.04	1.99% ± 0.08	1.99% ± 0.07	1.98% ± 0.09	1.91% ± 0.14
10	2% ± 0.03	1.99% ± 0.10	1.95% ± 0.13	1.92% ± 0.15	1.81% ± 0.16

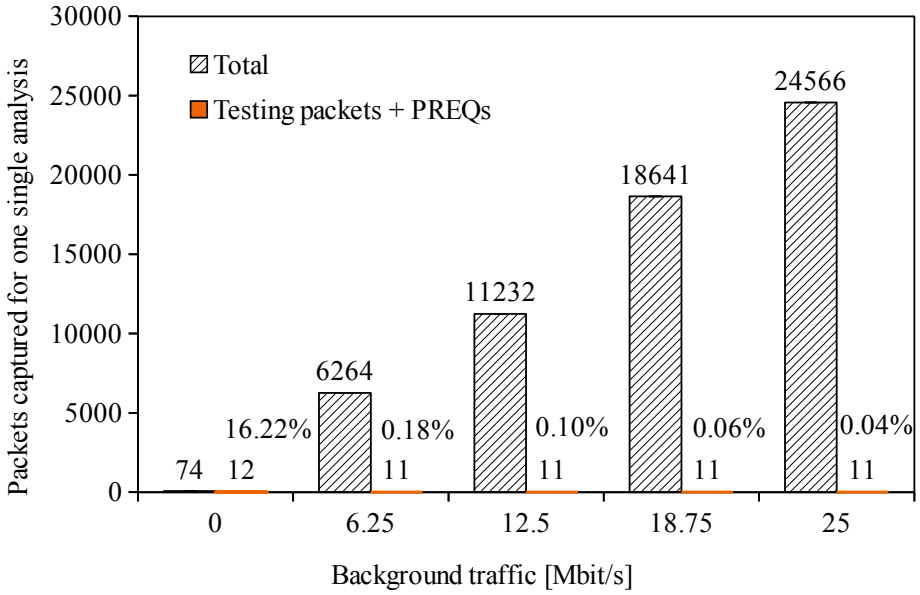
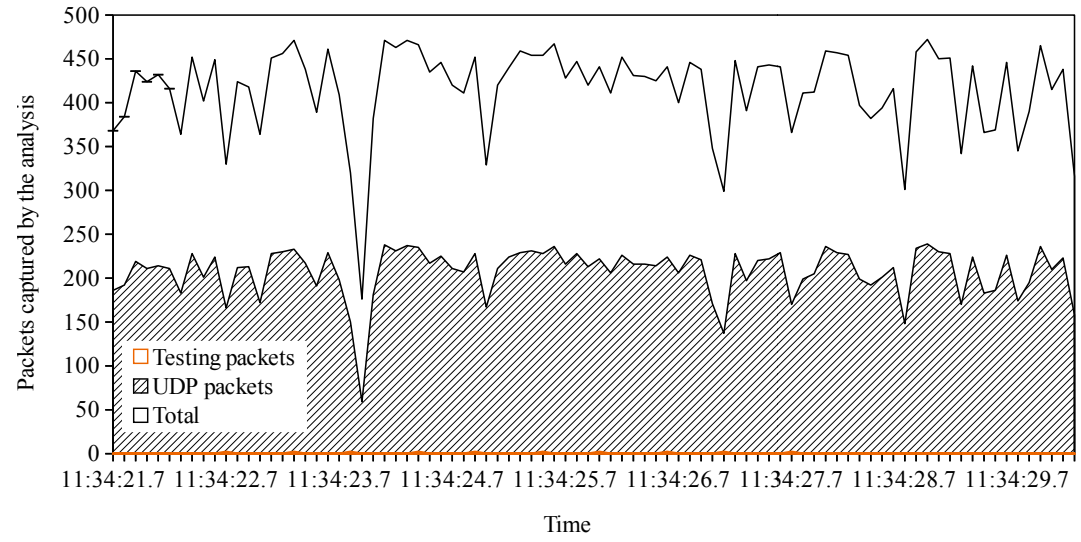
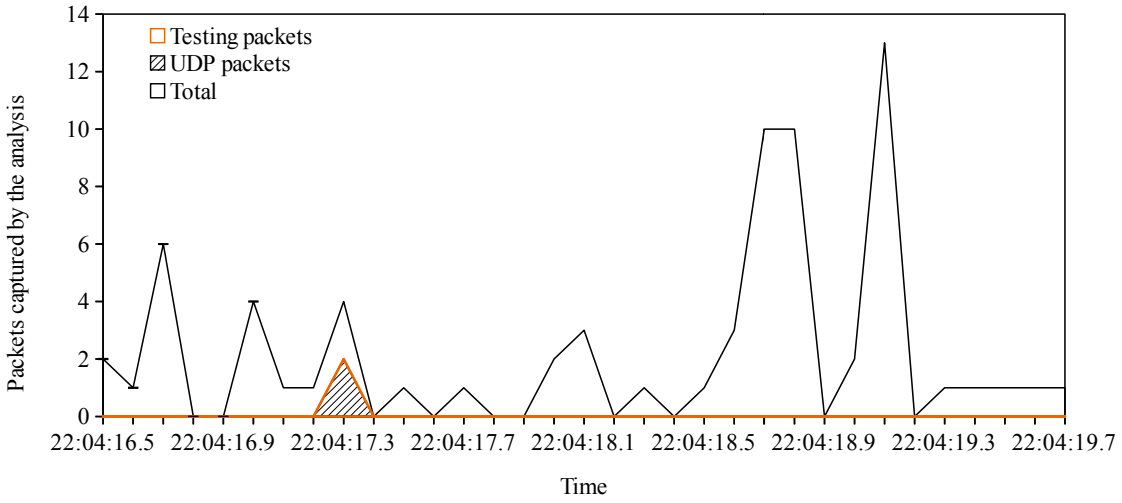


Figure 29: Comparison between the amount of testing packets vs. the total number of packets present in the data gathered for one single analysis. The percentage values over the red bars represent the percentage of testing packets from the total.



(a) Trace of the traffic gathered by DogoIDS for one single analysis when the background traffic was 25 Mbit/s. The number of testing packets is not zero but comparatively smaller than the UDP packets and total traffic.



(b) Trace of the traffic gathered by DogoIDS for one single analysis when no background traffic was present.

Figure 30: Two traces of the traffic gathered for two analyses.

testing packets (plus a PREQ) represent 0.18% of the total traffic. With maximum background traffic, 25 Mbit/s, the testing packets are 0.04% of the total traffic.

To demonstrate, we depict the traces of the data gathered for two different analyses. Figure 30a shows a trace for a background traffic of 25 Mbit/s, and Fig. 30b depicts a trace when there was no background traffic present. In Fig. 30a, almost half of the traffic can be attributed to UDP packets, because we transmit UDP packets to generate background traffic. A very tiny part corresponds to the AP-NIDS testing packets and control packets. *We set the Time To Live (TTL) of the PREQ transmitted from the AP-NIDS to 2, so that they do not unnecessarily propagate through the network*. In Fig. 30b, the testing packets are the only UDP packets present. The rest of the traffic can be attributed to control traffic.

We argue that the network overhead generated by DogoIDS is not high. Only when there is no background traffic in the network, the testing packets, of course, represent much more of the total traffic than with background traffic present. However, in Fig. 26 we show that for no background traffic, sending only 2 testing packets already throws a low false positive rate. The AP-NIDS could be configured to be aware of the network traffic and automatically self-adjust the number of testing packets transmitted. The IDSes in the literature are not mobile, but deployed on many or all the nodes of the network, and they need to exchange data between the nodes in order to detect attacks collaboratively. Comparing the traffic injected by our AP-NIDS and the distributed IDSes of the literature is not entirely fair, because the detection principle of our AP-NIDS is fundamentally different and must inject traffic in order to unveil the presence of the attacks.

5.6 SUMMARY

In this chapter, we have studied how to control important parameters of the active probing process and therefore improve the detection quality. One of the most important conclusions obtained is that we should replicate the testing packets transmitted to prevent misinterpretations due to packet loss. If control packets are involved in the active probing process, they should be replicated as well. The goal of this chapter was to pragmatically identify the problems of active probing and propose solutions to minimize them. We aimed to identify the most general problems or parameters of an active-probing-based IDS. To show them in practice, we resorted to our specific testbed. Of course, the active probing process is general and can be implemented in any context (different networks, protocols), therefore the specific adjustment depends on the context used. For example, in vehicular networks we might increase the replications more than we did for our experiments in a mesh network. Protocols other than IEEE 802.11s require different control packets.

6 MODELING PROBE SELECTION AND ATTACK INFERENCE

> Things should be made as simple as possible — but no simpler.
>
> Albert Einstein

This chapter proposes a Bayes model for attack inference and an algorithm for probe selection to minimize the number of probes executed.

6.1 INTRODUCTION

In the previous chapters, we have witnessed how the active probing mechanism transmits testing packets into a wireless network in order to unveil malicious activity. The testing packets are transmitted within the context of a probe, and each probe is intended to provide information about one particular attack (see Fig. 4 in Chapter 4). An active-probing-based IDS is, consequently, provided with a set of probes large enough to cover the wide spectrum of existing attacks. A priori, it might be thought that in order to detect a particular attack, the complete set of probes has to be executed. Executing the complete set is, at the same time, the easiest way of implementing and designing an AP-NIDS. Nevertheless, this approach yields some serious disadvantages: it increases the overall duration of the active-probing analysis, and it increases the network overhead, because more testing packets are transmitted.

Rish et al. proposed a mechanism for probe selection for fault determination in distributed systems [91, 92]. Their idea is similar conceptually to what we need for AP-NIDS, but it is meant for determining faults in complete systems, as a result of probing their individual components (servers). Their components can be in states "up" or "down", where the state of a system is inferred from the state of all its components. In an AP-NIDS, however, the scenario is different. The probings performed are reduced to single individual nodes, where their state is not only "up" or "down", but all the different possible attacks. In this work, we reformulate their approach for being applicable to active probing intrusion detection. Our approach is, however, valid for other applications where individual nodes have to be probed against different states, i.e. generally applicable.

In this chapter, we propose a mechanism that frees us of executing the complete set of probes in order to detect the attacks [93]. In other words, we propose a mechanism to minimize the set of probes to execute. As we already saw in Fig. 8 in Chapter 4, we designed an "Inference Engine" module for the architecture of active-probing-based network intrusion detection systems. In this chapter, we fill this block. We model a temporal-selective Bayesian classifier to infer the state of a network node under test. This allows us to classify if a node is misbehaving or not, based on the outcome of a set of active probes. To implement this classifier, we design a recursive probe selection scheme. It is based on the current posterior of the Bayes classifier and a prediction step and facilitates reducing the number of active probes while maximizing the insights gained by the set of probes executed.

This chapter is structured as follows: We start in Section 6.2 by introducing the design of our temporal-selective Bayesian classifier, including its definitions and mathematical model. We also introduce our algorithm for probe selection and we present a numerical example. In Section 6.3 we describe the experiments we performed with our AP-NIDS and its setup, and we present and discuss the results in Section 6.4. Finally, Section 6.5 summarizes and closes this chapter.

6.2 TEMPORAL-SELECTIVE BAYESIAN CLASSIFIER

In this section, we describe the temporal-selective Bayesian classifier to optimize the active probing process. We first concisely describe the basics of the naive Bayes classifier, which we then extend to the temporal domain. We further introduce a novel feature selection strategy for our classifier. To illustrate the intuition behind our approach, we provide a numerical example afterward.

6.2.1 *Definitions*

Let $\mathcal{N} = \{n_i\}_{i=0}^{I}$ be a set of $I+1$ mutually exclusive class labels/hypotheses, which are in our case the different attacks to be detected by the AP-NIDS. The class label n_0 represents "normal condition" and stands for the detection result "no attack detected". Let $N_x \in \mathcal{N}$ be the discrete state of the target node at position x in the network that is analyzed by the AP-NIDS. And let $\Phi = \{\phi_j\}_{j=1}^{J}$ be a set of binary features $\Phi \in \{\text{true}, \text{false}\}^J$. In our case, each feature ϕ_j equals the result of a probe j applied to the target node and is designed to detect one specific attack n_i. From now on, we assume there is only one specific probe per attack and a probe exists for each attack. The probe j corresponds to attack n_i if $i = j$. The different attacks to be investigated by the AP-NIDS are for example n_1 "dropping of ARP packets" or n_2 "dropping of DNS packets".

6.2.2 *Results of the probes*

Before continuing with the Bayes classifier, let us clarify the outcome of a probe. For our Bayesian model, a probe is a black box that transmits some testing packets, it performs some steps, and it returns a binary value $\Phi \in \{\text{true}, \text{false}\}^J$. In our model, we understand the result "true" as successful, i.e. the probe detected the attack as it was designed to.

Internally, a probe consists of analyzing the data gathered after transmitting the testing probes. Based on these data, the internal logic of a probe determines if the probe returns true or false. There is no restriction on the internal logic of a probe. For exemplary purposes, we summarize the internal logic used for the concrete case of our AP-NIDS.

We programmed the internal logic of the probes based on *tests* and *attack patterns*. The term *test* refers to inquiries made to the data gathered after the transmission of the testing packets. The tests are matched against the data gathered, employing some matching function, and the results compared to pre-defined knowledge of malicious activity, or attack patterns. The internal logic of our deployment in simple pseudo code is as depicted in Algorithm 1.

Algorithm 1: Pseudo code of the internal logic of the probes in our deployment.

```
Input: Probe to launch (P)
Output: Outcome of the probe
SendTestingPackets();
Data ⟵ GatherDataFromNetwork();
TestResult ⟵ DoTest(Data,P);
if TestResult==Pattern then
    return true;
end
return false;
```

We show some of the lists of tests employed in our deployment. Please note that the given lists are not (and cannot be) exhaustive. Examples of the set of tests for the analysis of the MAC header and the Mesh Control field include:

- ▷ Testing packet forwarded?

- ▷ Testing packet payload modified?
- ▷ Testing packet reappeared?
- ▷ Address 1 modified?
- ▷ Address 2 modified?
- ▷ Address 3 modified?
- ▷ Address 4 modified?
- ▷ Address 5 modified?
- ▷ Address 6 modified?
- ▷ Mesh TTL decreased?
- ▷ Mesh Flags changed?
- ▷ Mesh Sequence Number modified?
- ▷ Mesh Sequence Number increased?

And examples of the set of tests for the analysis of the HWMP elements are:

- ▷ Path discovery started?
- ▷ PREQs forwarded?
- ▷ PREP received?
- ▷ PERR received?
- ▷ PERR Destination Address forged?
- ▷ Element ID modified?
- ▷ Hop Count modified?
- ▷ Originator MSTA Address forged?
- ▷ Mesh Sequence Number modified?
- ▷ Target Address crafted?
- ▷ Lifetime modified?
- ▷ Metric modified?
- ▷ Flags field modified?

6.2.3 *Bayesian model for attack detection*

Our goal is to infer whether the target node is attacked or not and — in case of attack — which kind of attack has happened. The brute force way of detecting an attack would be to apply all probes to the target node, look at all the feature vectors and decide for the probe that answers "true". This simple method has several problems. First, what to decide if several probes lead to a positive answer. This usually happens because it is typically not possible to design probes that lead to an unambiguous classification result. Second, we have to execute all probes, which causes a large amount of traffic, is computationally expensive, and time consuming. We propose pursuing a probabilistic detection approach to reduce this overhead. Now, the AP-NIDS provides conditional probabilities $p(\phi_i|N_x)$ for a specific feature of a

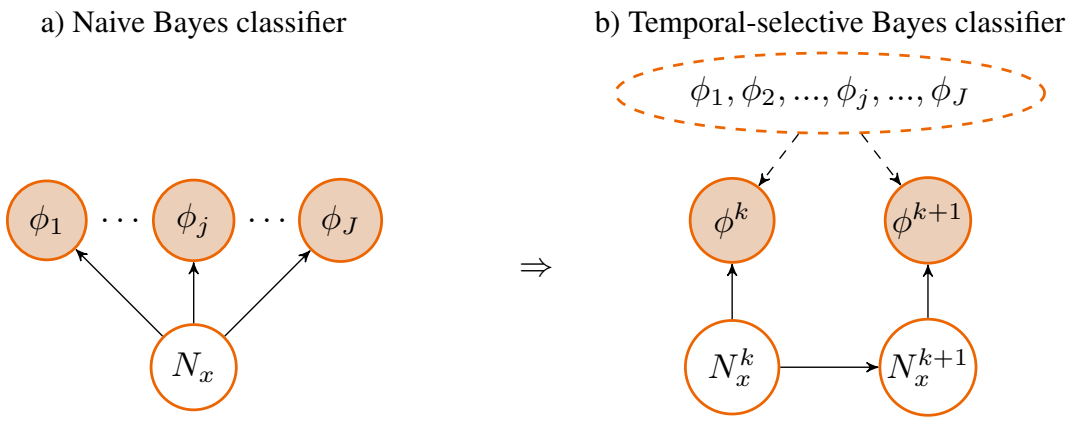


Figure 31: a) Graphical model of the naive Bayes classifier and b) the proposed temporal selective Bayes classifier. N_x^k is the state of the target node x and ϕ^k is the selected probe at timestep k.

probe ϕ_i given a specific attack $N_x = n_i$. This makes it possible to tell how uncertain the AP-NIDS is for its feature given a specific attack. Here, $p(\phi_i|N_x)$ is a discrete conditional probability that can be summarized in a CPT with probabilities $p(\phi_i = \text{true}|N_x = n_i) = p_{ti}$ and $p(\phi_i = \text{false}|N_x = n_i) = p_{fi}$ for all possible attacks $n_i \in \mathcal{N}$ and binary feature values $\phi_i \in \{\text{true}, \text{false}\}$ shown in Table 5.

Table 5: Conditional Probability Table $p(\phi_i|N_x)$

ϕ_i	n_0	n_1	n_2	n_3	n_4	n_5	...	n_i	...	n_I
true	p_{t0}	p_{t1}	p_{t2}	p_{t3}	p_{t4}	p_{t5}	...	p_{ti}	...	p_{tI}
false	p_{f0}	p_{f1}	p_{f2}	p_{f3}	p_{f4}	p_{f5}	...	p_{fi}	...	p_{fI}

Each probe results in a certain CPT that has to be learned from data or constructed empirically. It satisfies $p(\phi_i|N_x) \geqslant 0$ and $\sum_{\phi_i} p(\phi_i|N_x) = 1$. Assuming that the features ϕ_i of the different probes are statistically independent $p(\Phi|N_x) = \prod_i p(\phi_i|N_x)$ and introducing a prior probability $p(N_x)$, we are able to infer the posterior probability $p(N_x|\phi_i)$ for each probe individually using Bayes' rule:

$$p(N_x|\phi_i) \propto p(\phi_i|N_x)p(N_x)\,. \tag{5}$$

Combining all individual features to one common posterior probability, we arrive at the naive Bayes classifier [94] as depicted in Fig. 31 a):

$$p(N_x|\Phi) \propto p(N_x)\prod_{j=1}^{J} p(\phi_j|N_x) \tag{6}$$

This classifier fuses all the different information sources Φ, namely the features of the different probes, to one common output. Which attack $n_\star$ the AP-NIDS assumes to be correct can be found by maximizing the posterior:

$$n_\star = \arg\max_i p(N_x = n_i|\Phi)\,. \tag{7}$$

Using this approach, we are able to express uncertainty about the features and the classification result and consider the statistical dependency between attacks. In addition, we are able to

account for prior knowledge modeled with the prior $p(N_x)$ preferring specific attacks. But still, all probes have to be taken into account. Therefore, we unroll the naive Bayes classifier along discrete time and only allow for one probe $\phi^k \in \Phi$ to be executed at each time step k. This leads to a first order Markov chain [94] for the state of the target node as depicted in Fig. 31 b). Hence, the state of the target node N_x^k also becomes time dependent and we arrive at a time dependent posterior

$$p(N_x^{k+1}|\phi^{1:k+1}) \propto p(\phi^{k+1}|N_x^{k+1}) \underbrace{\sum_{N_x^k} p(N_x^{k+1}|N_x^k) p(N_x^k|\phi^{1:k})}_{p(N_x^{k+1}|\phi^{1:k})}, \tag{8}$$

whereas $\phi^{1:k}$ is the set of all probes $\phi^{1:k} = (\phi^1, \phi^2, ...\phi^k)$ executed up to time k. This Markov chain is completely given by the initial prior probability $p(N_x^0)$, the transition probability $p(N_x^{k+1}|N_x^k)$ and the observation likelihood $p(\phi^k|N_x^k)$ applying one specific probe $\phi^k = \phi_i$ at each time step k. Assuming the transition probability to be the identity matrix, there is no uncertainty introduced because of the state transition and for the temporal prior/prediction it follows $p(N_x^{k+1}|\phi^{1:k}) = \sum_{N_x^k} p(N_x^{k+1}|N_x^k) p(N_x^k|\phi^{1:k})$. Hence, the Markov chain equals a stepwise executed naive Bayes classifier. Additional sources of information can easily be introduced, e.g. knowledge about correlations between attacks that have already been detected in neighboring nodes and the attack to be detected in the current node. However, up to now, we do not know which probe ϕ_i to execute at which time step k. It thus remains open: *how many probes to execute and in which temporal order to execute them to yield a reliable detection result?*

6.2.4 *Optimal probe selection*

To identify which order of probe execution is most suitable, we define an optimization criterion that is based on the current posterior and a prediction step. Note that we allow each probe to be executed only once. This leads to a recursive selection mechanism that is optimal with respect to the optimization criterion. The criterion has to be evaluated anew at each time step and results in the next best probe to be executed.

To answer the question on the minimal number of probes to execute, we define a simple stopping criterion that is based on a threshold on the maximum of the posterior. Once the maximum posterior probability is higher than the threshold, the AP-NIDS stops probe execution and provides the final classification result.

More precisely, we define two sets of probes with their elements changing over time. The first set $\Gamma^- := \phi^{1:k}$ is the set of all the probes executed up to timestep k. The second set $\Gamma^+ := \Phi - \Gamma^-$ is the set of the remaining probes that can still be executed in the future.

Our selection mechanism is a two step procedure that is executed each time after a new posterior has been calculated. First, the current posterior is maximized following Equation 7

$$n_\star^k = \arg\max_i p(N_x^k = n_i|\Gamma^-). \tag{9}$$

Now, the attack $n_\star^k$ the AP-NIDS assumes to be most probable is time dependent. If this maximum probability exceeds a given threshold θ at time k

$$p(N_x^k = n_\star^k|\Gamma^-) > \theta, \quad \Rightarrow \quad n_{\text{final}} = n_\star^k, \tag{10}$$

the AP-NIDS is certain enough about its classification result, the temporal selective naive Bayes classifier stops and the final decision is n_{final}. If Equation 10 is not fulfilled the AP-NIDS selects a new probe ϕ^{k+1} to be launched based on the following optimization criterion:

$$\phi^{k+1} = \arg\max_j \{p(N_x^{k+1} = n_\star^k|\Gamma^-, \phi_j)\}_{j\in\Gamma^+} \tag{11}$$

Algorithm 2: Probe selection and state update algorithm

Input: State N_x^k of the target node
Output: Updated state N_x^{k+1} of the target node
$\Phi \longleftarrow$ available set of probes;
$\Gamma^- \longleftarrow \emptyset$;
while $n_\star^k < \theta$ **and** $\Gamma^+ \neq \emptyset$ **do**
  $n_\star^k \longleftarrow$ attack to be most probable (Equation 7);
  Select $\phi^{k+1} \in \Gamma^+|$ so that after executed, $n_\star^k$ is still maximum (Equation 11);
  Execute ϕ^{k+1};
  $\Gamma^- \longleftarrow \Gamma^- \cup \phi^{k+1}$;
  Update N_x^{k+1} (Equation 8);
end

This criterion analyzes the next possible posterior probabilities

$$p(N_x^{k+1} = n_\star^k | \Gamma^-, \phi_j) = p(N_x^{k+1} | \Gamma^-) p(\phi_j | N_x^{k+1} = n_\star^k)$$

that could happen. Note that we do not depend on the outcome of the probe (true or false). We loop for the complete set of remaining probes Γ^+ and select the one that maximizes the prediction 11 of the new posterior. This is somehow intuitive, because we seek an affirmation of the current attack estimate that results in an increase in posterior probability in order to exceed the threshold in Equation 10. Since the remaining probes are not designed for testing the current attack estimate, the posterior only increases if the new probe also holds some information about the current attack (encoded in its CPT), although it favors a different one.

We map the above reasoning into Algorithm 2 and give a numerical example to illustrate the working of the algorithm.

Selection of the threshold θ

We left the selection of the threshold θ to empirical and/or subjective reasoning of the administrator. For our experiments in this chapter, we selected the value $\theta = 0.40$. It should be noted that choosing a high θ implies low values for the rest of the states of the vector. Practically, this means that the outcome of the CPTs should give a very accurate and high certainty about a certain state, and this can be difficult to achieve. Choosing a low θ reduces the sensitivity of the inference, which can lead to a false classification.

The selection of the first probe

Special care has to be taken in the selection of the first probe, since the choice of the first probe can have a huge impact on the number of probes to be launched afterward. For this kind of initialization, as much prior knowledge about the process as possible should be involved.

In our experiments, the first probe to be launched is determined by the system administrator. Here the experience of the administrator is needed — the administrator is assumed to know which attack is more common to occur in the network where the AP-NIDS is deployed. Practical alternatives to choose the first probe could be:

- ▷ Choosing very critical probes first: the probe that is designed to detect the attack that is most harmful for the actual deployment of the network.
- ▷ Choosing very certain probes first: the probe that offers a maximal probability in the CPTs $p(\phi_i | N_x)$ for one specific attack.
- ▷ Choosing more likely probes first: the probe that maximizes the prior probability $p(N_x^0)$.

Table 6: Probes implemented for detecting selective dropping of different services

Probe	Reference	Type of packets dropped	State
ϕ_1	ARP	ARP Requests and Responses	n_1
ϕ_2	DNS	UDP packets with port 53	n_2
ϕ_3	HTTP	TCP packets with port 80	n_3
ϕ_4	HTTPS	TCP packets with port 443	n_4
ϕ_5	SSH	Every packet with port 22	n_5
ϕ_6	TCP	Every TCP packet	n_6
ϕ_7	UDP	Every UDP packet	n_7
ϕ_8	All	Every UDP and TCP packet	n_8
	—	Normal condition (no dropping)	n_9

Of course, a tradeoff between all three criteria could also be chosen: A probe that is 1) quite critical, 2) quite certain to detect and 3) quite likely to occur.

Implementation of the CPTs

The conditional probability tables are basically built on statistical analysis of data and/or logical reasoning. If the environment is known and we have a large database, we can learn the CPTs from the database, which is a process known as batch learning [95]. If statistical information cannot be learned, we can build the CPTs based on logical reasoning (if "X" is true, how possible is it that "Z" is true?). As an example, to perform the experimentation of this dissertation, we configured our AP-NIDS with the set of probes described in Table 6, and the conditional probability tables associated with them are shown in Table 7. We built them using logical reasoning. We assigned high values if the probes succeeded for the attack they were created for. In Table 7 this coincides with a diagonal starting at the first probe $p(\phi_{ARP}|N)$. The rest of the probabilities are reasoned. Note that the values of the tables (columns) are normalized before being injected into the Bayes model, so that the high values written are smoothed afterwards. In practice, this means that choosing 99% probability does not imply the final inference on that state will be automatically high or definitive.

6.2.5 *Numerical example*

Let us clarify the above probe selection algorithm with an example. We will consider the active probing of a node which its state at the time k is represented by N_x^k. The possible classes for the state of the node are $\{n_1, ..., n_9\}$. n_1 stands for "dropping of ARP packets", n_2 for "dropping of DNS packets", and so on and so forth, so that they fit the probes described in Table 6.

We assume an equal distributed prior probability $p(N_x^k)$:

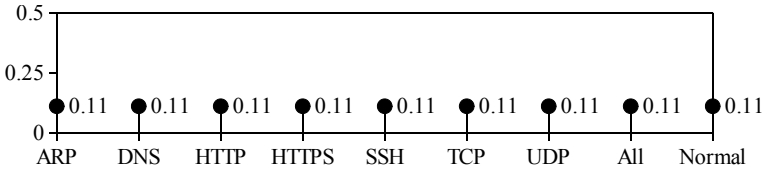
$$p(N_x^k) = (0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11)$$

Let us assume that the starting probe is $\phi^k = \phi_3$. The probe yields true and this result is linked with the corresponding CPT (see Table 7). The updated state N_x^k results from (5):

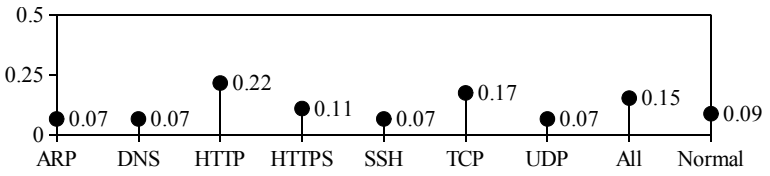
$$N_x^k = (0.07, 0.07, 0.22, 0.11, 0.07, 0.17, 0.07, 0.15, 0.09)$$

Table 7: Conditional probability tables implemented for our experimentation

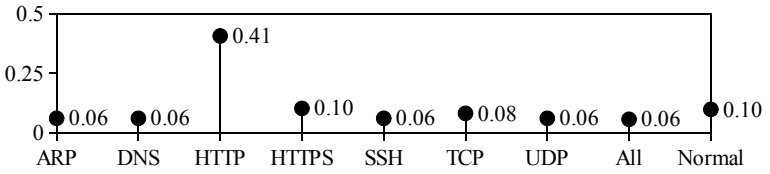
CPT	ϕ_i	ARP	DNS	HTTP	HTTPS	SSH	TCP	UDP	All	Normal
$p(\phi_{ARP}\|N)$	true	0.99	0.20	0.50	0.30	0.40	0.30	0.50	0.50	0.40
	false	0.01	0.80	0.50	0.70	0.60	0.70	0.50	0.50	0.60
$p(\phi_{DNS}\|N)$	true	0.20	0.90	0.50	0.30	0.30	0.30	0.90	0.50	0.40
	false	0.80	0.10	0.50	0.70	0.70	0.70	0.10	0.50	0.60
$p(\phi_{HTTP}\|N)$	true	0.70	0.30	0.99	0.50	0.30	0.80	0.30	0.70	0.40
	false	0.30	0.70	0.01	0.50	0.70	0.20	0.70	0.30	0.60
$p(\phi_{HTTPS}\|N)$	true	0.60	0.60	0.60	0.99	0.60	0.60	0.60	0.60	0.01
	false	0.40	0.40	0.40	0.01	0.40	0.40	0.40	0.40	0.99
$p(\phi_{SSH}\|N)$	true	0.50	0.40	0.50	0.40	0.99	0.50	0.50	0.50	0.30
	false	0.50	0.60	0.50	0.60	0.01	0.50	0.50	0.50	0.70
$p(\phi_{TCP}\|N)$	true	0.50	0.50	0.01	0.50	0.50	0.75	0.50	0.80	0.40
	false	0.50	0.50	0.99	0.50	0.50	0.25	0.50	0.20	0.60
$p(\phi_{UDP}\|N)$	true	0.50	0.30	0.50	0.50	0.50	0.30	0.99	0.80	0.40
	false	0.50	0.70	0.50	0.50	0.50	0.70	0.01	0.20	0.60
$p(\phi_{All}\|N)$	true	0.50	0.50	0.50	0.50	0.50	0.30	0.40	0.99	0.40
	false	0.50	0.50	0.50	0.50	0.50	0.70	0.60	0.01	0.60



(a) Starting vector. We use an equally distributed prior probability.



(b) Updated state after launching the probe ϕ_3. HTTP has the highest probability.



(c) Final state after launching probe ϕ_6. The node is dropping HTTP packets.

Figure 32: Sequence of the updated state values when detecting the dropping of HTTP packets.

The state with the highest probability so far is $n_\star^k = n_3 = 0.22$ ("dropping of HTTP packets"). The launched probe is stored in $\Gamma^- = \{\phi_3\}$. According to our algorithm, the process should continue until the belief in one of the states is higher than a threshold θ. For this example and for the experimentation in this dissertation, we set $\theta = 0.40$. The AP-NIDS continues the process and selects the next probe according to our algorithm. This step involves looping for the set of remaining probes $\Gamma^+ = \{\phi_1, \phi_2, \phi_4, ..., \phi_8\}$ according to Equation 11. For space reasons we omit the calculation of determining which probe gives more information about n_3 and we assume this probe is $\phi^{k+1} = \phi_6$ ("dropping of TCP packets"). The probe results in false and the new value for N_x^{k+1} is:

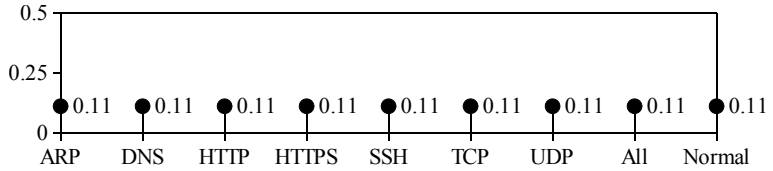
$$N_x^{k+1} = (0.06, 0.06, 0.41, 0.10, 0.06, 0.08, 0.06, 0.06, 0.10)$$

The certainty of $n_\star^{k+1} = n_3$ increases to 0.41, and it is greater than θ. The AP-NIDS concludes that the state of N_x is n_3 = "dropping of HTTP packets". The sequence of the updating process of the vector N_x^k is shown in Fig. 32.

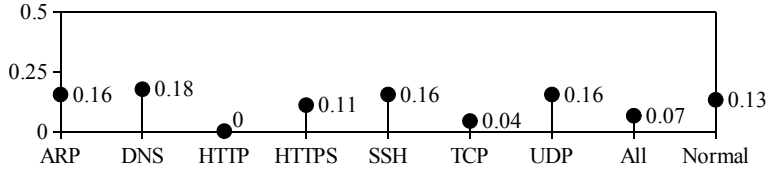
In Fig. 33 we depict another example of the state updating sequence: the attack that takes place is the "dropping of UDP packets", and the starting probe is ϕ_3.

6.3 EXPERIMENTS

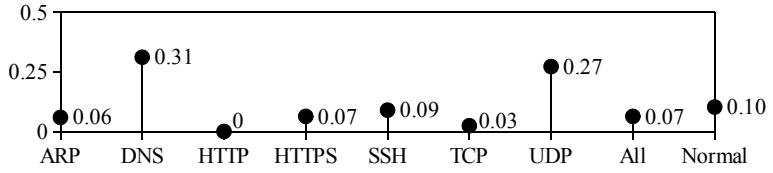
The goal is to experimentally demonstrate that using the proposed Bayesian-based Inference Engine has notable advantages in comparison to not using an Inference Engine, but running the complete set of probes. First, we want to examine how the number of probes executed to detect a particular state decreases when using our Inference Engine. Consequently, we expect that the overall detection time also decreases. Secondly, we examine if using our Inference Engine, on the other hand, leads to a higher false positive rates. Finally, we analyze the effects on the network overhead.



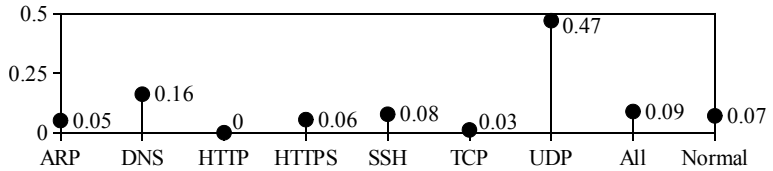
(a) Starting vector. We use an equally distributed prior probability.



(b) Updated state after launching the probe ϕ_3 (it yielded false). DNS has the highest probability.



(c) Updated state after launching the probe ϕ_2 (it yielded true). DNS still has the highest probability.



(d) Final state after launching probe ϕ_7 (it yielded true). The node is dropping UDP packets.

Figure 33: Sequence of the updated state values when detecting the dropping of UDP packets.

6.3.1 *Setup*

For our experiments we use an indoor wireless mesh testbed composed of 16 fixed nodes and 2 netbooks running the AP-NIDS. We use the configuration of two interfaces for an AP-NIDS (see Section 5.2). To implement different probes, we use the selective dropping of packets attack. The target node of our AP-NIDS launches this attack in their different modalities: *Dropping of ARP*, *Dropping of HTTP*, *Dropping of HTTPS*, *Dropping of only TCP*, *Dropping of only UDP*, *Dropping of TCP and UDP*, *Dropping of DNS*, *Dropping of SSH*, *Dropping of all packets*, and *No dropping*. In addition, we perform the experiments with varying background traffic. We transmit UDP traffic between the target node and a neighboring node with the following values: 0 Mbit/s (0%), 6.25 Mbit/s (25%), 12.5 Mbit/s (50%), 18.75 Mbit/s (75%), 25 Mbit/s (100%). The last one represents the maximum achievable throughput that we obtain in our network (100%). The technical details and description of the testbed is detailed in Section A.3 of the Appendix on page 108.

6.4 RESULTS

6.4.1 *Number of probes launched and detection times*

As we show in Fig. 34, it was not necessary to launch all the probes to detect the attacks. Even when the node was not dropping packets (Normal), only 5 of the 8 probes implemented were launched. In the best case, for detecting dropping of HTTP packets, only two probes

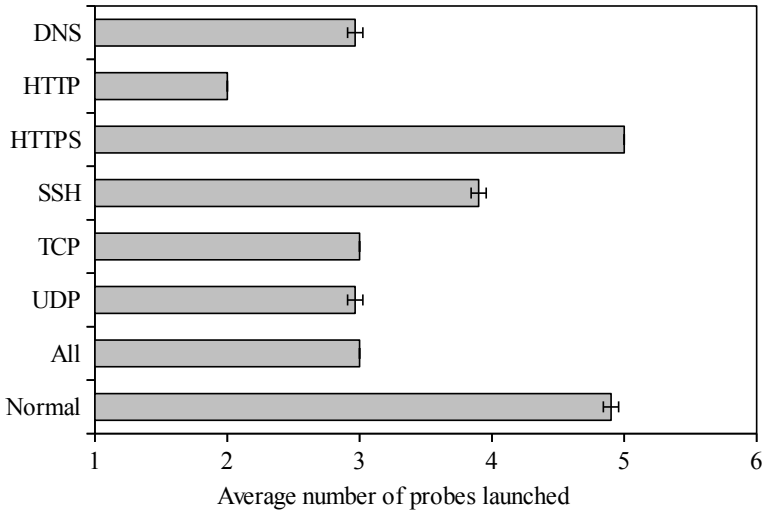


Figure 34: Number of probes needed to be launched for detecting different attacks.

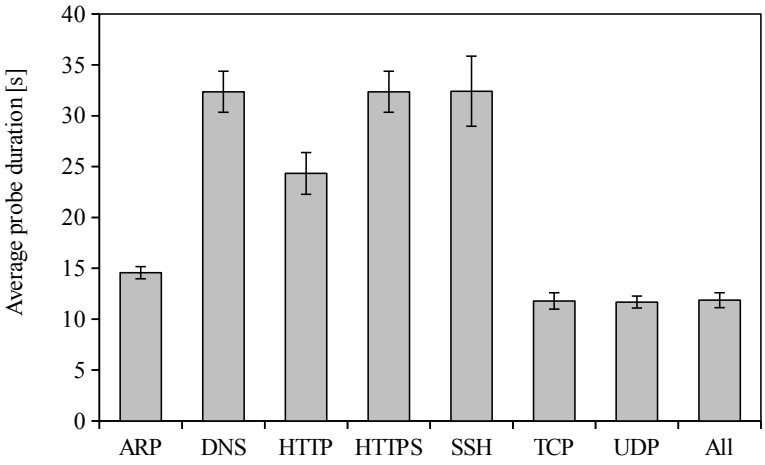


Figure 35: Average total duration of the individual probes.

were necessary, which represents 25% of the set of probes. If the AP-NIDS did not have an algorithm for selecting the probes, it would need to transmit the complete set of probes in order to detect the attack precisely, given that the success of one probe is not enough to alert the attack with confidence.

Figure 35 shows the average duration of the execution of each probe. These times include the transmission of the probe plus the gathering of data afterward. Intermediate times such as pauses between probes and processing times are not considered, because they do not belong to the individual probes but to the analysis.

As we can observe, the duration of the probes differs for each probe. Some probes demand about 11 s, while others around 30 s. If we calculate the average across all the probes, we get a probe duration of 21.42 s $\pm$1.53. It is easy to observe the time saved when using the probe selection algorithm. For each probe that is not executed, the analysis requires 21.42 s less on average. This is an important saving, if we consider that the AP-NIDS is set with many different probes.

Figure 36 depicts the average duration of the complete active probing process. As we already observed in Fig. 35, some attacks need more time to be detected because more probes are launched. The duration of the active probing is slightly affected by the traffic load of the network. In Table 8, we show the analysis duration for detecting the dropping of SSH packets, and for detecting the dropping of UDP packets, when the network is loaded with different levels of traffic.

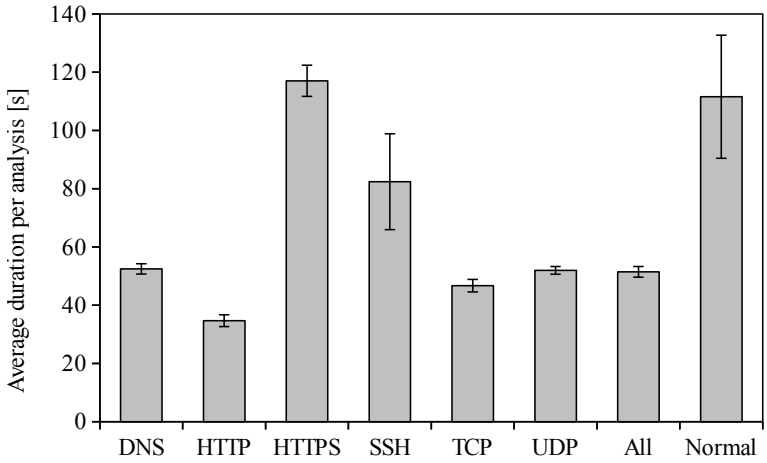


Figure 36: Average total duration of the detection of the different attacks.

Table 8: Average analysis duration for different background traffic levels

Background traffic	**Analysis duration [s]**
Attack launched: dropping of SSH packets	
0 Mbit/s	87.76 ± 6.83
6.25 Mbit/s	101.84 ± 26.33
12.5 Mbit/s	109.54 ± 15.07
18.75 Mbit/s	101.74 ± 28.52
25 Mbit/s	107.01 ± 28.57
Attack launched: dropping of UDP packets	
0 Mbit/s	51.32 ± 3.80
6.25 Mbit/s	65.55 ± 11.38
12.5 Mbit/s	66.66 ± 10.20
18.75 Mbit/s	61.28 ± 8.96
25 Mbit/s	64.54 ± 6.52

Table 9: Results of the attack detection. SSH and UDP are additionally measured with varying background traffic.

Attack	**True positives**	**Well classified**	**Ill classified**
DNS	100%	93%	7%
HTTP	100%	100%	0%
HTTPS	100%	100%	0%
SSH	100%	93%	7%
TCP	100%	100%	0%
UDP	100%	97%	3%
All	100%	100%	0%
Normal	93%	93%	7%
Background traffic	**True positives**	**Well classified**	**Ill classified**
Dropping of SSH packets			
6.25 Mbit/s	100%	83%	17%
12.5 Mbit/s	100%	93%	7%
18.75 Mbit/s	100%	80%	20%
25 Mbit/s	100%	90%	10%
Dropping of UDP packets			
6.25 Mbit/s	100%	97%	3%
12.5 Mbit/s	100%	100%	0%
18.75 Mbit/s	100%	93%	7%
25 Mbit/s	100%	100%	0%

6.4.2 *Attack detection*

The results for the detection of attacks are pretty satisfactory. We differentiate between the true positives and well classified attacks for the following reason: to validate our Inference Engine, well classified is an important metric to examine how well/poorly it is able to classify the attacks. However, for intrusion detection, it is more important to detect an attack in the first place. Ultimately, if the attack is detected but ill classified, an alarm is still generated and countermeasure actions can be taken. According to the results described in Table 9, we observe that our Bayes classifier had minimal troubles classifying some attacks. In no case do well classified attacks fall under 93%, when there was no background traffic present.

Additionally, we are also interested in observing what happens with our Bayes classifier where background traffic is present. In Table 9 we depict the results for varying background traffic when detecting the attacks dropping of SSH packets, and dropping of UDP packets. The background traffic does affect attack detection. In the worst case, we observe that the dropping of SSH was 80% well classified. The reason for the bad classification is the following. To detect the dropping of SSH packets, for example, the Inference Engine launches four different probes (see Fig. 34). If one of these probes gives an incorrect result, the Bayes classifier makes false presumptions about the probability of one of the possible states. It unleashes a set of wrong decisions of the probe selecting algorithm and, eventually, a wrong inference about the attack. For this reason, improving the attack classification implies improving the "robustness" of the

individual probes. This is, however, not a drawback of the Bayes classifier. If a probe fails, even when transmitting the complete set of probes, the IDS can get erroneous detection.

6.4.3 *Overhead*

Our goal is to measure the impact of the active probing on the network overhead. DogoIDS transmitted in the maximum case 1.85 Kbit/s for a complete analysis. That was the case of HTTP, including the control packets and testing packets generated. We argue that an AP-NIDS doesn't harm the network overhead. For example, our testbed handles 25 Mbit/s in the best case. The throughput generated by DogoIDS represents only 0.01% of the maximum achievable throughput. For 25% utilization (6.25 Mbit/s), the throughput of DogoIDS represents 0.03%. Of course, the amount of data generated by DogoIDS depends on how many probes are executed and how many bytes are injected per probe. The data generated by the probe is a "user-defined" parameter. For example, in our experiments we generate "lightweight" testing packets, which contain a very small payload. We consider this is an appropriate choice, but in the end, AP-NIDS administrators are free to create their testing packets as they consider. However, regardless of the individual testing packets, with our Bayes classifier and probe selector we directly reduce the number of injected bytes by reducing the number of probe launches.

On limiting the probe set to the available bandwidth

We affirm that an AP-NIDS hardly causes any harm to the network overhead. However, if used for the case of very specific applications (bandwidth-limited sensor networks, for example), we propose a different approach to probe selection. An interesting approach is to limit the set of probes or attacks to detect to a certain allowed throughput. For example, if the policies of the network establishes that only 5% of the maximum network throughput can be used for active probing intrusion detection, we can configure our AP-NIDS to not exceed that limit. That can be done by limiting the probe set, reducing the length of the testing packets, or increasing the intervals between probes or testing packets. We did not implement this idea but we leave it as future work.

6.5 SUMMARY

In this chapter, we highlighted two contributions. We have modeled a temporal-selective Bayes classifier to infer the state of a device under test. Different to similar classifiers proposed in the literature, the state of a device is a vector of all possible malicious behaviors. The transmission of probes feeds the Bayes classifier and updates the state. A recursive probe selection scheme, based on a prediction step, facilitates reducing the number of probes while maximizing the insights gained. We have modeled our classifier in general and applied it to solve a particular problem, the malicious state inference for an active-probing-based network intrusion detection system for wireless multihop networks. We have implemented our model and performed testbed experimentation. We have showed how an inference model can improve the reduction of the number of probes executed and therefore reduce overhead and detection time.

7 EVALUATION

> More people are killed every year by pigs than by sharks, which shows you how good we are at evaluating risk.
>
> Bruce Schneier

> It doesn't matter how beautiful your theory is, it doesn't matter how smart you are — if it doesn't agree with experiment, it's wrong.
>
> Richard P. Feynman

In this chapter, we put the active-probing intrusion detection on the field. We implement three different attacks in our testbed and we test the performance of our AP-NIDS.

7.1 INTRODUCTION

In this chapter, we combine the contributions from Chapters 4, 5 and 6. We use the lessons learned from the past chapters, which led us understand how to deploy and set up our active-probing based network intrusion detection system.

We implement three different attacks, the *selective dropping of packets*, the *black hole attack*, and the *colluding misrelay attack*. As described in Section 2.2 (on page 8), these are representative attacks with different difficulties. The selective dropping of packets is quite powerful. It is easy to implement, which makes it attractive for the average attacker, and it can severely harm the network performance if it is strategically executed. Its detection is not difficult and many intrusion detection systems can detect it (as we observed in Section 2.3). The black hole attack is also a powerful attack and its detection is a bit more complex. To detect it, the IDS should interact with the attacker and correlate some information. The colluding misrelay attack corresponds to the attacks which are hard to detect, since the position of the IDS node plays an essential role. We analyzed in Section 2.3 that most of the IDS present in the literature cannot detect this kind of attack, or if they do, they are pure theoretical systems. In this chapter, we show that the active-probing intrusion detection is practical for detecting attacks of all classes introduced in Chapter 2.

This chapter is organized as follows. Section 7.2 describes the setup for the evaluation of this chapter based on the lessons learned in the previous chapters. We also describe our testbed and evaluation goals in Section 7.2. Sections 7.3, 7.4 and 7.5 describe, evaluate and present the results for the detection of the selective dropping of packets, of the black hole attack and of the colluding misrelay attack, respectively. We discuss the results in Section 7.6 and conclude the chapter in Section 7.7.

7.2 EXPERIMENTAL SETUP

The previous chapters were necessary to first understand how to configure an active-probing-based network intrusion detection system. In Chapter 5, we measured the effects of the repetition of testing packets on the detection quality and false positives. We saw how transmitting replications of testing packets is beneficial to some extent. We showed in Section 5.5.2 on page 42, that with five repetitions of testing packets, our AP-NIDS achieved

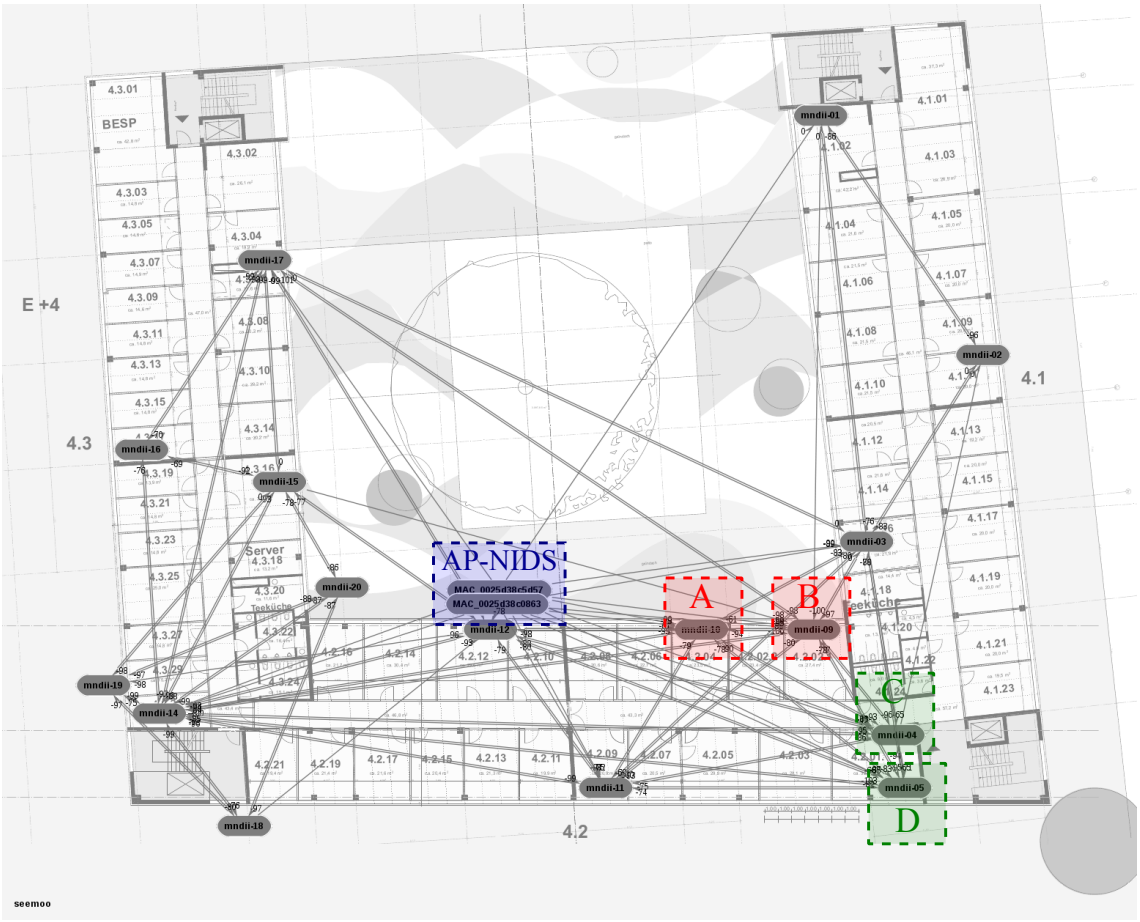


Figure 37: Setup of our mesh testbed and a snapshot of the network topology. The location of the AP-NIDS nodes does not represent their exact physical location (they are located inside the room left of node A).

satisfactory results. We also studied the intervals between repetitions with varying background traffic levels. An interval of 500 ms allowed us not to overload saturated nodes, yet obtain low false positives and detection rates (Section 5.5.3 on page 49). In addition, we investigated how bad links between the AP-NIDS and target node lead to wrong detections (Section 5.5.1 on page 42). In the experiments of this chapter, we analyze target nodes with RSSI greater than -70 dBm. In Chapter 6 we designed a Bayesian engine for attack inference and probe selection. We did extensive experimentation with variations of the selective dropping of packets. From this chapter, we learned that using our statistical model helps us to reduce the number of probes executed, yet accurately detect (classify) attacks. In this chapter we extend the probe set of DogoIDS, adding two completely different attack detection rules, the black hole and the colluding misrelay attack.

7.2.1 *Testbed setup*

We follow the same setup as in the previous chapters. We highlight the key parameters of our testbed and the detailed description can be found in Section A.4 of the Appendix on page 113. We employ an indoor testbed composed of 16 mesh nodes installed on two floors of an office building. The layout of the testbed is shown in Fig. 37. Node A and node B are attacker nodes, and nodes C and D are background traffic generators. We configured our AP-NIDS with the following probes: *Black hole*, *Colluding misrelay*, *Dropping of ARP*, *Dropping of HTTP*, *Dropping of HTTPS*, *Dropping of only TCP*, *Dropping of only UDP*, *Dropping of TCP and UDP*, *Dropping of*

DNS, *Dropping of SSH*, *Dropping of all packets*, and *No dropping*. We send 5 repetitions of testing packets each with intervals of 500 ms. We perform the experiments with varying background traffic. We transmit UDP traffic between the target node and a neighboring node with the following values: 0 Mbit/s (0%), 6.25 Mbit/s (25%), 12.5 Mbit/s (50%), 18.75 Mbit/s (75%), 25 Mbit/s (100%). The last one represents the maximum achievable throughput that we obtain in our network (100%).

7.2.2 *Goals*

The main goal of this evaluation is to demonstrate that our active-probing technique can detect real attacks. We aim to extend the probes of the Bayes classifier to two more attacks with different levels of complexity. We intend to measure detection rates and detection times for different attacks. In addition, we want to test our intrusion detection system with the parameters set in the past chapters and also using different background traffic levels. We also want to measure how the intrusion detection system works in terms of CPU usage and memory consumption with the new attacks.

7.3 DETECTING THE SELECTIVE DROPPING OF PACKETS/SERVICES

The selective dropping of packets is the attack that we have extensively used in the first chapters of this dissertation. In Chapter 6, we implemented the dropping of packets for eight different services or cases. For this reason, we do not measure this attack again, since the environment and configurations would not change. The parameters to configure the AP-NIDS were already performed in Chapter 5 and applied in Chapter 6 and in this chapter. Nevertheless, we will discuss the results obtained for these attacks in this chapter.

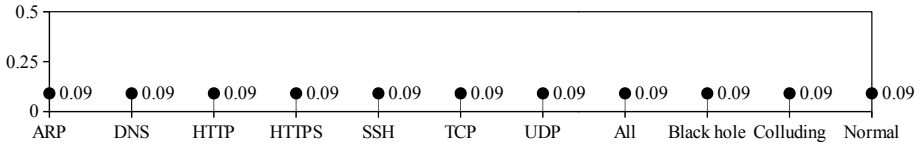
7.4 DETECTING THE BLACK HOLE ATTACK

We implemented the black hole attack in our wireless mesh testbed. We set one of the nodes of our network as the attacker, running the black hole attack. The attack basically interprets path requests and replies to them announcing the attacker node as the best route to the intended destination. And when data packets are sent to the attacker node, it drops them. We performed the experiments with varying background traffic. We transmit UDP traffic between the target node and a neighboring node with the following values: 0 Mbit/s (0%), 6.25 Mbit/s (25%), 12.5 Mbit/s (50%), 18.75 Mbit/s (75%), 25 Mbit/s (100%). For the technical details and description of the testbed, refer to Section A.4 of the Appendix on page 113.

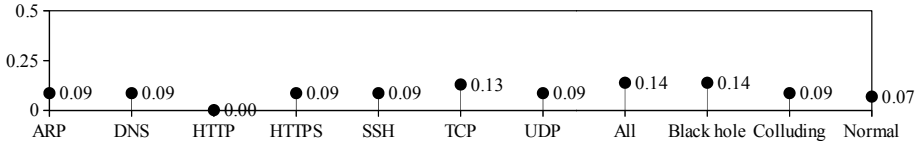
The first step towards detecting the black hole attack is transmitting testing packets with the content of a PREQ element to the target node. The packets are limited in Mesh TTL to two hops so that they do not spread. The MAC addresses announced in the Target Address field of the PREQs are randomly crafted, so under no attack situation, responses are not expected to be received (the chances of choosing a MAC address which corresponds to one mesh station of the network are low.) On the contrary, if after the test PREPs are received, the AP-NIDS has a notable indicator to suspect the presence of the black hole attack. Then, the AP-NIDS sends testing packets to the target node and if they are dropped, the AP-NIDS concludes that the node is launching this attack.

In a nutshell, the AP-NIDS performs these steps:

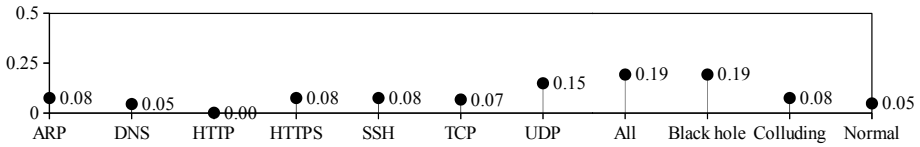
- The AP-NIDS sends a PREQ to the target node asking for the path to a node with a random MAC address.
- In presence of the attack, the target node announces that it has the best route to that node.
- The AP-NIDS sends testing packets to the target node.



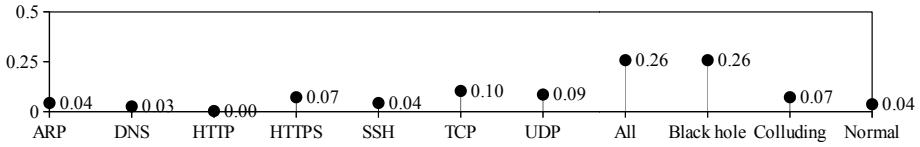
(a) Starting vector. We use an equally distributed prior probability.



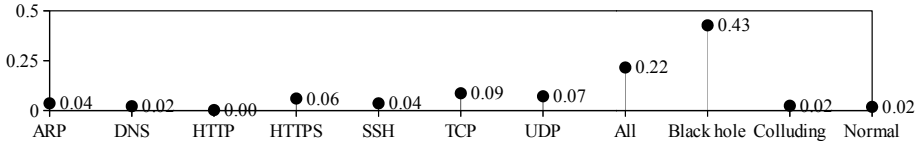
(b) Updated state after launching the first probe, *Dropping of TCP*. All and black hole have the same probability.



(c) Updated state after launching the probe *Dropping of UDP*. All and black hole have the same probability.



(d) Updated state after launching the probe *All*. All and black hole still have the same probability.



(e) Final state after launching probe *Black hole*. The black hole attack is now detected.

Figure 38: Sequence of the updated state values when detecting the black hole attack.

▷ The testing packets are dropped by the target node and the AP-NIDS detects the attack.

We add the probe to detect the black hole attack, and the colluding attack, to the probes already used for Chapter 6. We do not change the starting probe of the AP-NIDS used for Chapter 6, the Dropping of TCP packets. Leaving this probe is good example for showing the dependence between attacks. The target node running the black hole attack does drop packets, so all the probes aiming to detect dropping of packets (Dropping of UDP, Dropping of TCP, etc.) return a positive value (attack detected). These probes give some probability to the black hole attack in our Bayesian engine, and the black hole attack is just detected when the corresponding probe for detecting it is launched. In Fig. 38 we show the sequence of the belief updating process.

7.4.1 *Results*

Table 10 depicts the detection rate, number of probes launched, and the duration of each analysis for the black hole detection. The detection rate is quite satisfactory, the attack was detected for all the different values of background traffic. It is possible to achieve 100% detection rate for the black hole attack because the detection of the attack is quite particular and the AP-NIDS does not confuse it with other attacks. Only in the case of black hole, the attacker node answers to the specially crafted PREQs and then drops the data packets. The

Table 10: Results of the black hole attack detection.

Background traffic	Detection rate	Probes launched	Analysis duration [s]
0 Mbit/s	100%	4	49.34 ± 0.42
6.25 Mbit/s	100%	4	50.14 ± 0.42
12.5 Mbit/s	100%	4	50.81 ± 0.46
18.75 Mbit/s	100%	4	51.96 ± 0.46
25 Mbit/s	100%	4	51.85 ± 0.57

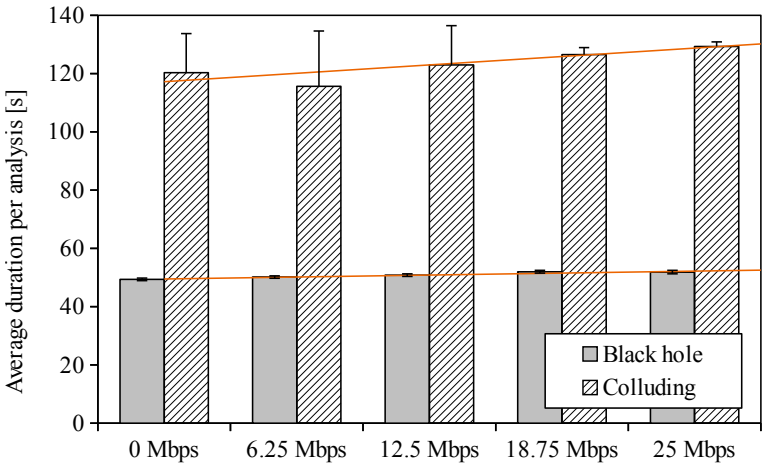


Figure 39: Average total duration of each analysis for black hole and colluding attack detection.

Table 11: Average CPU usage and average memory load of DogoIDS when detecting the black hole attack.

CPU usage				
0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
91.27% ± 0.33	90.45% ± 0.43	88.63% ± 6.49	88.61% ± 6.06	88.01% ± 6.21
Memory load				
0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
2.28% ± 0.04	2.29% ± 0.03	2.27% ± 0.11	2.23% ± 0.13	2.25% ± 0.09

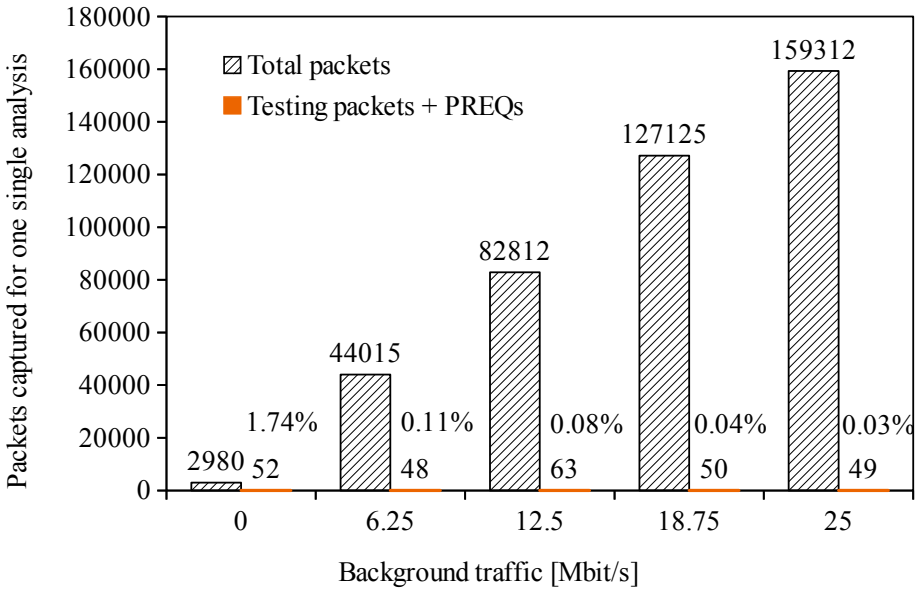


Figure 40: Comparison between the amount of testing packets vs. the total number of packets present at the data gathered for one single analysis. The percentage values over the red bars represent the percentage of testing packets out of all probes.

background traffic does not affect the detection rate. Only the analysis duration is slightly affected, it has a trend to increase the duration when increasing the background traffic. This is easily observable in Fig. 39. However, they are only two seconds between no saturated network and completely saturated network.

In Table 11 we depict the measurements of the CPU usage and memory load of DogoIDS for this experiment. These measurements were performed when DogoIDS was running the active detection, i.e. per analysis. DogoIDS can handle the background traffic properly. The memory load is stable for the different levels of background traffic. The hardware of the IDS node is not badly affected by our proof-of-concept AP-NIDS. We remark that we use an Asus Eee PC 1005HA netbook to run DogoIDS, which has 1 GB RAM and a low-power processor. If in the future, DogoIDS has to perform more complex calculations to detect other attacks, more powerful hardware can be employed. The IDS node is only used for intrusion detection, so that its resources are fully available to DogoIDS.

Regarding the network overhead, we depict in Fig. 40 the number of packets for five different analyses, each one for a different value of background traffic. We used a similar plot in Section 5.5.4 of Chapter 5. We observe that the network overhead, in terms of packets, introduced by DogoIDS is minimal. Only when there is no background traffic present, the testing packets represent around 1.74% of all packets.

Table 12: Results of the colluding attack detection.

Background traffic	**Detection rate**	**Probes launched**	**Analysis duration [s]**
0 Mbit/s	97%	7.07	120.30 ± 13.39
6.25 Mbit/s	90%	6.80	115.63 ± 18.97
12.5 Mbit/s	93%	7.03	122.92 ± 13.49
18.75 Mbit/s	100%	7.03	126.48 ± 2.41
25 Mbit/s	100%	7	129.30 ± 1.55

7.5 DETECTING THE COLLUDING MISRELAY ATTACK

We implement the colluding misrelay attack in our testbed. Two nodes run the attack. One of them forwards packets to the second one, and the second one drops them. The goal of this attack is that the transmitter of packets cannot realize that the attack is taking place, because packets are being dropped 2 hops away from it. This is why this attack is hard to detect, the position of the intrusion detection system plays a fundamental role here.

To detect the attack, our AP-NIDS follows the following reasoning. We distinguish between two cases. In the first case, the AP-NIDS is in the neighborhood of node A, but not of node B. The detection follows:

- Node A and node B are colluding. Node A forwards all the traffic to B, and B drops it.
- AP-NIDS sends testing packets to the target node A. Since the testing packets are addressed (as final destination) to the second interface of the AP-NIDS, and this interface is a direct neighbor of node A, the packets should be forwarded from node A to the interface of the AP-NIDS.
- But, the target node forwards the testing packet to the colluding node B. This is a clue for the AP-NIDS. The AP-NIDS moves closer to node A and B.
- The AP-NIDS sends testing packets to B and observes that it drops them. The attack is alerted.

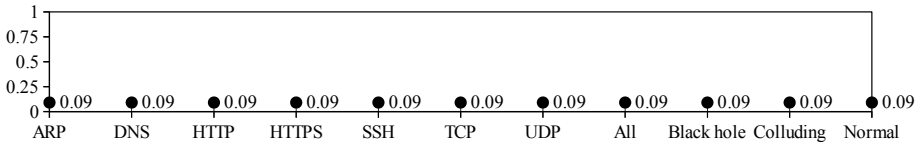
In the second case, the AP-NIDS is in the vicinity of A and B, and alerts that A forwards all packets to B. The detection is:

- Node A and node B are colluding. Node A forwards all the traffic to B, and B drops it.
- The AP-NIDS sends testing packets to B and observes that it drops them. The attack is alerted.

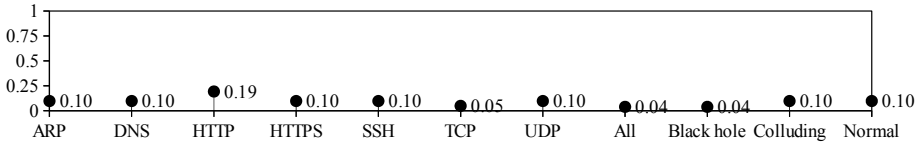
For our evaluation, we skip the first case, moving the AP-NIDS in a pre-detection phase, since we want to evaluate if the attack is finally detected or not (second case). Again, we make use of the Bayesian model designed in the previous chapters. In order to show that it works, we show in Fig. 41 the sequence of the belief updating process.

7.5.1 *Results*

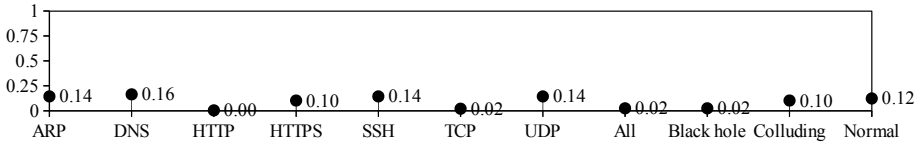
Similarly to the black hole attack, in Table 12 we depict the detection rate, number of probes launched, and the duration of each analysis for the colluding misrelay attack detection. For this attack, DogoIDS did not achieve 100% detection rate in every case, but over 90%. Given the complexity of the attack and of the attack detection, we consider theses values quite satisfactory. The analysis duration takes more time than for detecting the black hole attacks



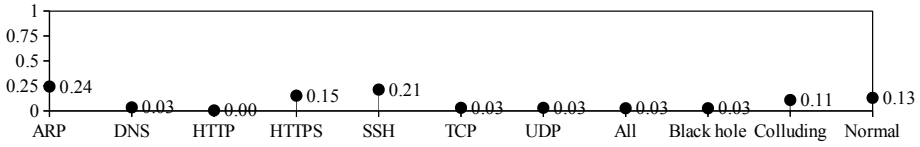
(a) Starting vector. We use an equally distributed prior probability.



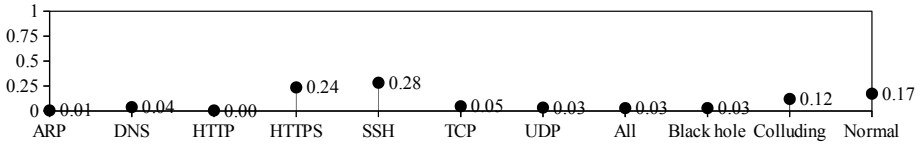
(b) Updated state after launching the first probe, *Dropping of TCP*.



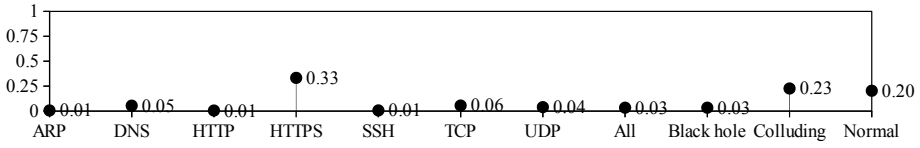
(c) Updated state after launching the next probe, *Dropping of HTTP*.



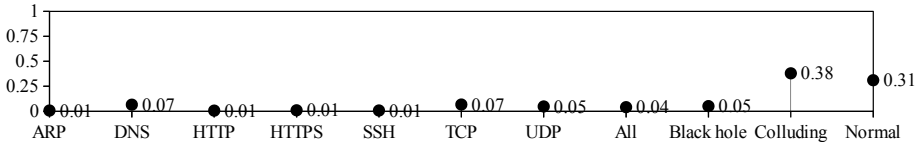
(d) Updated state after launching the probe *Dropping of DNS*.



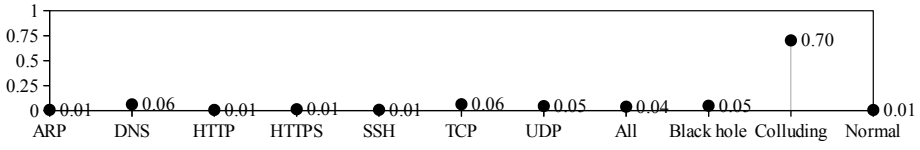
(e) Updated state after launching the probe *Dropping of ARP*.



(f) Updated state after launching the probe *Dropping of SSH*.



(g) Updated state after launching the probe *Dropping of HTTPS*.



(h) Final state after launching probe *Colluding*. The colluding attack is now detected.

Figure 41: Sequence of the updated state values when detecting the colluding attack.

Table 13: Average CPU usage and average memory load of DogoIDS when detecting the colluding attack.

CPU usage				
0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
93.33% ± 3.32	92.59% ± 4.67	93.06% ± 0.16	92.97% ± 4.67	93.10% ± 0.27
Memory load				
0 Mbit/s	**6.25 Mbit/s**	**12.5 Mbit/s**	**18.75 Mbit/s**	**25 Mbit/s**
2.30% ± 0.02	2.30% ± 0.03	2.30% ± 0	2.30% ± 0.03	2.30% ± 0

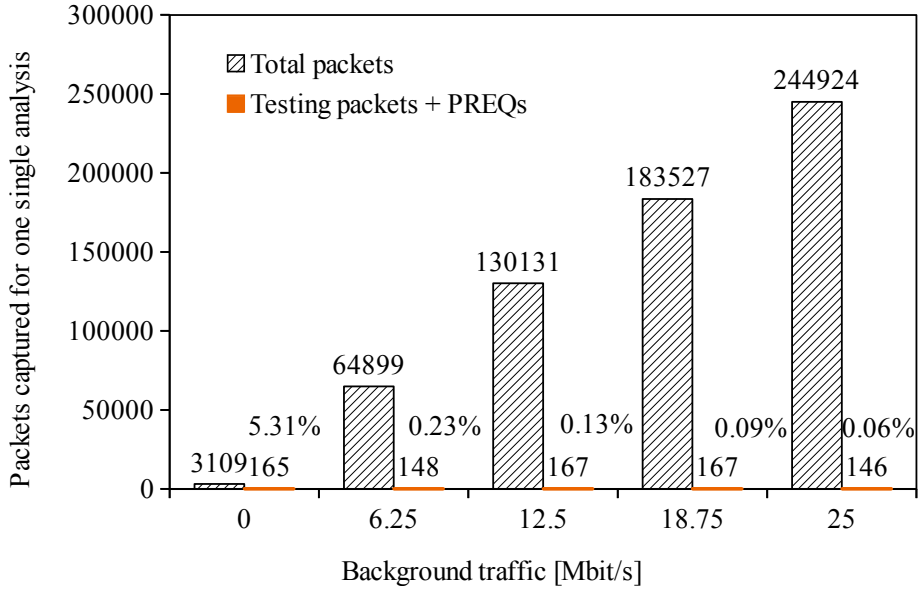


Figure 42: Comparison between the amount of testing packets vs. the total number of packets present at the data gathered for one single analysis. The percentage values over the red bars represent the percentage of testing packets from the total.

because more probes are executed. Additionally, detecting the colluding attack involves "two active probing" processes. Testing packets are sent to node A, analyzed, and then testing packets are sent to node B. The detection time in total might be long. However, our goal is to show that by means of active probing, this attack can be detected at all. In practical environments, it will depend on the network/security policies if this attack has to be detected or not. For example, if the network administrator wishes to detect attacks quickly, an AP-NIDS node can be deployed and the rule for detecting the colluding misrelay attack deactivated. A second IDS node can be also deployed, with the goal of detecting this complex attack with longer detection times.

In Table 13 we depict the CPU usage and memory load of our proof-of-concept DogoIDS. The values are quite stable for the different background traffic levels. Similarly to the detection of the black hole attack, the hardware of the IDS node is not strongly affected by our AP-NIDS.

Regarding the network overhead, we depict in Fig. 42 the number of packets for five different analyses, each one for a different value of background traffic. The network overhead, in terms of packets, introduced by DogoIDS is minimal. Of course it is bigger than by detecting the black hole attack, since more probes are launched, but it is still low. Only when there is no background traffic present, the testing packets represent around 5.31% of all packets. In the other cases, it is under 1%.

7.6 DISCUSSION

The first important and trivial conclusion of our evaluation is that we were able to detect the attacks we aimed to and with acceptable results. The selective dropping of packets was detected at a rate of over 93% (see Table 9 on page 67), the black hole attack at a rate of 100% and the colluding misrelay attack at a rate of over 90%. We are satisfied with these results. We can conclude that our active-probing technique is able to detect these attacks and we demonstrated it in a practical environment. With the help of one intrusion detection node running an AP-NIDS, we were able detect simple and complex attacks in a 16 node network, while these network nodes were not disturbed at all. We demonstrated that our implementation is lightweight in network overhead. We measured that the traffic generated by DogoIDS was under 6% of the total traffic when there was no background traffic present. With background traffic, the generated traffic by DogoIDS was in some cases less than 0.1%. These are important results because we can argue that introducing a minimal overhead on the network nodes, as in distributed intrusion detection systems, our AP-NIDS can detect complex attacks with a minimal usage of network traffic. As we discussed in Section 6.4.3 on page 68, even if the network traffic is a problem, an AP-NIDS can be configured to limit its detection capabilities if it reaches a given value of network overhead. And this is not a limitation but a design choice. The detection times are in general long, ranging from below 40 s and up to 130 s for all the attacks. The detection time has two faces in our dissertation. The first one is that the active-probing technique by design is slower than locally and passively monitoring data. The active-probing IDS has to first transmit testing packets, then listen to the target node, and finally analyze the gathered information. This process is longer than continuously inspecting local information. The other face is that we implemented DogoIDS as a proof-of-concept, but for time reasons, we did not optimize every detail of the implementation. For sure, there are times that can be minimized in a professional implementation of an AP-NIDS. But, to conclude, we are satisfied that we were able to detect the attacks. In a real deployment, the detection times might depend on the security requirements. For example, if the security requirements are strict and some attacks, like the dropping of packets, have to be detected in few seconds, passive and active intrusion detection systems can be combined. Passive IDS might introduce overhead on the network nodes, but it can detect dropping of packets quickly. An active IDS can look for more complex attacks which passive IDS cannot.

7.7 SUMMARY

In this chapter, we have experimentally demonstrated that our proposed active-probing-based network intrusion detection system works in detecting the different classes of attacks, from the most simple ones to those more complex. The previous chapters helped us to understand how the active-probing technique works and how to correctly adjust it. We used the insights of the previous chapters to complete the puzzle of the active-probing-based intrusion detection in this chapter.

8 EXTENSION

> Creativity is just connecting things. When you ask creative people how they did something, they feel a little guilty because they didn't really do it, they just saw something. It seemed obvious to them after a while.
>
> Steve Jobs

In this chapter, we introduce a passive metric for attack inference that can be integrated to active probing as an extension to complement the attack detection.

8.1 INTRODUCTION

This chapter presents an extension to the active-probing-based network intrusion detection concept. The extension we present here is independent, but also connected to the rest of our work because we designed it as a kind of "add-on" to an AP-NIDS. We designed and tested it separately from the rest of the work since it is general and can also be applicable to other systems. We propose a metric to detect attacks based on the variations on the number of neighboring nodes.

Because of their complex nature, MANETs face two major challenges: on the one hand, the characteristics of these networks, such as the open transmission medium, allow for a wide range of new attacks compared to classical wired networks. The attacks are difficult to detect and/or mitigate by using the same methods as in wired networks [20]. The challenge is to find new techniques to accurately detect these attacks. On the other hand, due to the lack of centralized management, a trust relation should be established between two neighboring nodes in order to guarantee that data packets are forwarded until they reach the destination [96].

We tackle the first challenge and, viewed from a broader perspective, we propose a new metric that can contribute to the second challenge [97]. Our specific contributions of this chapter are:

- *A lightweight metric called Neighbor Variation Rate (NVR) for anomaly detection.* We study how the neighborhood varies over time and we represent it with a quantitative measurable value. We exploit the strong dependency on time and on the mobility of nodes and build a model for our metric (Section 8.3).
- *A detection model.* We create a detection model where we find the limits of the expressiveness of the NVR with respect to the mobility of the nodes and we apply it for anomaly/intrusion detection (Sections 8.1.1 and 8.4).
- *Validation and parametrization.* By means of extensive simulations of a MANET running AODV with the software ns-3, we validate and understand the fundamental parameters of our detection model (Section 8.5).
- *Real-world deployment and external dataset analysis.* In order to demonstrate the efficacy of our metric we build a proof-of-concept and deploy it in an IEEE 802.11s-enabled wireless mesh network composed of 23 nodes (18 static and 5 mobile). We utilize existing security tools to create realistic attacks in our testbed. We also examine our metric applied to an external dataset from a MANET composed of 30 fully mobile nodes running AODV (Section 8.5).

8.1.1 *Attacker model and assumptions*

The attacker model in this chapter differs slightly from the general one. We define the attacker model of this chapter as follows. We assume that there is at least one attacker and that the attacker is an insider, i.e. it has the respective authentication credentials and is a member of the network. The attacker is able to either 1) inject/remove nodes from the network, or 2) inject false routing information, e.g. by filling-up the routing tables with false data.

We assume that we only analyze the NVR of nodes not compromised. We remark that our main goal is to evaluate the accuracy of our metric and not to develop an intrusion detection system (our metric can be used for intrusion detection among other things). In the case of intrusion detection, if the system administrator suddenly turned off a big set of nodes or add a new set of nodes, the NVR would trigger an abnormality. The intrusion detection system could be aware of this sudden behavior, for example by differentiating those nodes that (based on their RSSI) suddenly appeared in the network from those who gradually left the neighborhood of a node.

8.2 RELATED WORK

A closely related study has been presented by Singh and Dutta [98]. By means of time series, they modeled the variation of the absolute number of neighbors of a certain node considering the time, transmission power, sampling periods and mobility models. With their model one can predict the number of neighbor nodes in future times under some assumptions. Another model for the prediction of the number of neighbor nodes has been introduced by Nguyen and Shinoda in [99]. Their model only considered the mobility according to the Random Waypoint Mobility model, which does not apply to many cases such as human mobility [100]. Our simulations use the Gauss-Markov mobility model and we perform real experimentation with humans walking. The work of Drugan et al. [101] estimates the number of neighboring nodes using the connectivity history.

The work presented in this chapter does not predict the number of neighbors at a certain point in time, but measures the variation of one-hop neighbors over time. Moreover, we investigate how the variation rate of neighbor nodes can help us identify anomalies rather than depend on the absolute number of nodes currently within transmission range.

8.3 NEIGHBOR VARIATION RATE (NVR)

In most of the literature, the definition of *neighbor* is only limited to the physical location of a node. In our work, we extend the definition of neighbor as follows: a node n_a is a neighbor node of n_b at the time t if *it is located* and *has been located* within the transmission range of n_b for a period of time long enough before t to let the routing protocol of n_b realize the existence of n_a. This condition is important when we consider the speed of nodes because a node moving too fast might not be detected in the vicinity of another node[1]. The transmission range is considered as a disk with a finite radius centered at the node. This assumption will be relaxed later and we will study it in the real world. All the nodes are located in a finite region of the Euclidean plane.

We want to determine a quantitative value that gives us the notion of how much the neighbors of a given node have changed from one point in time to another. The changes to which we refer are the addition and subtraction of nodes that are immediate (one-hop) neighbors and ready to communicate. In case of the IEEE 802.11s mesh protocol for example, we refer to the addition and subtraction of peer links. We are interested in the rate of this variation of neighbors rather than in the absolute quantity of nodes that have changed.

The reason behind the calculation of this rate is that we expect that a node can change its neighbors, i.e. add or lose nodes, from one moment to the next within a certain margin. As

1 For example, the open80211s implementation of IEEE 802.11s transmits beacons to detect nodes each second by default.

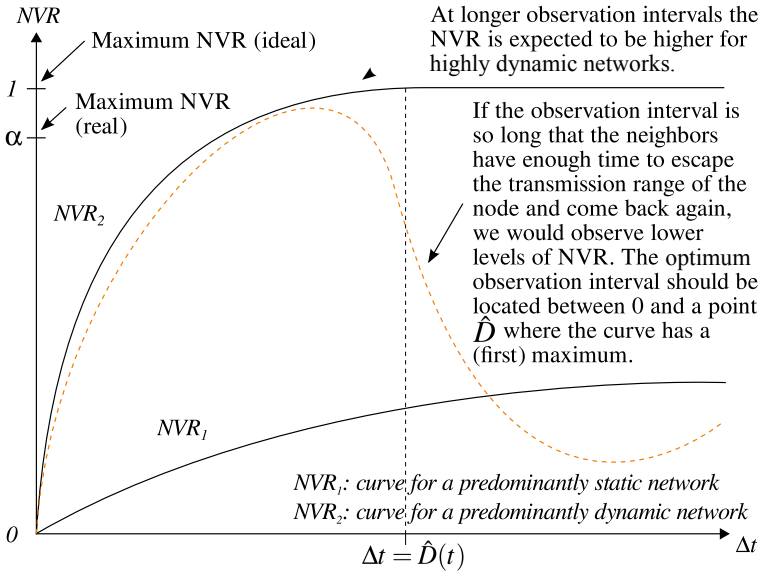


Figure 43: Conceptual representation of three curves of NVR as a function of the observation interval. The dashed line represents hypothetical behavior of the NVR when the observation interval is considerably long.

an example, if a node had 20 neighbors at a certain point in time, and 10 seconds later all of its neighbor nodes changed, this situation can be the result of an attack injecting/removing nodes. Alternatively, the node could be moving at a relatively high speed and the protocol is fast enough to handle these changes.

We define the NVR of a node n_k at the time t as a function $NVR(t) : \mathbb{R} \to \mathbb{R} \in [0, 1]$:

$$NVR(t) \triangleq \begin{cases} 0 & N(t) = N(t') = 0 \\ 1 - \dfrac{N(t) - W(t, t')}{\max(N(t), N(t'))} & \text{otherwise} \end{cases} \tag{12}$$

where $N(t)$ is the number of neighbors of the node n_k at time t, $N(t')$ is the number of neighbors of n_k at a previous time t':

$$t' = t - \Delta t \tag{13}$$

and $W(t, t')$ is the number of *new* neighbor nodes of the node n_k at time t in comparison to time t'.

Let us clarify these variables with an example: A node n_k announced to have 20 neighbor nodes at time t'. In the next measurement at time t it announces 30 nodes, and 13 of them were not present at the previous measurement at t'. The corresponding values are $N(t) = 30$, $N(t') = 20$, and $W(t, t') = 13$. From (12), this leads to an NVR of 0.85. Note that if $N(t) > N(t')$ then $W(t, t') \geqslant (N(t) - N(t'))$.

When the NVR takes a value of zero it indicates that there were no changes in the neighborhood (neither added nor removed nodes). At the other extreme, a value of 1 indicates that all the neighbors have changed.

8.3.1 *NVR in static and dynamic networks*

In a network where the nodes are primarily static in terms of physical location, we would expect to observe few changes to the neighbors of a node over a long period. On the contrary, in a network which is highly dynamic, like a vehicular network, we would expect that over

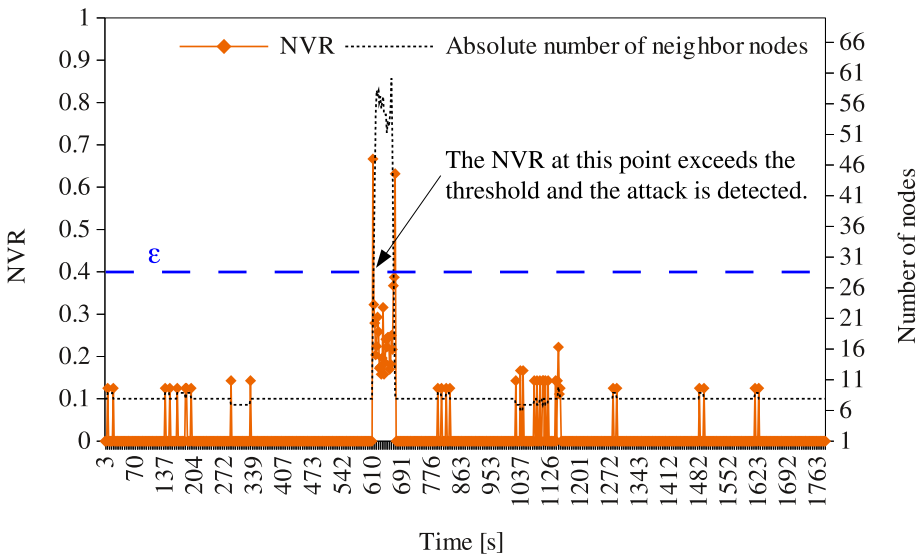


Figure 44: Example of the detection model. A node on our test-bed is attacked at a given point in time (between seconds 610 to 667).

a long period every node would have changed all its neighbors. This concept is depicted in Fig. 43. We call Δt the *observation interval*. It is the time between two consecutive readings on the number of neighbor nodes as defined in (13). As shown in Fig. 43, in highly dynamic networks we will have big variations in neighbor nodes if the observation interval is longer. Note that the curves do not necessarily converge to 1 because if the observation interval is considerably long and the mobile nodes move within an area but without escaping it, at some point in time each node will meet the same nodes it has met at previous measurements again, resulting in a low neighbor variation rate. This should be prevented by adopting low values for the observation interval. We will suggest a suitable value for Δt in Section 8.4.3. Throughout this chapter we will employ the terms *static* and *dynamic* to refer to the physical mobility of the nodes in a network.

8.4 DETECTION MODEL

We define a threshold that we will use to distinguish between a normal and an abnormal situation. We expect that the instantaneous changes of NVR(t) should not exceed a threshold $\varepsilon(\Delta t)$. In other words, given a network and nodes with certain characteristics (more/less mobility of nodes) we define a maximum value for NVR(t). This threshold depends on the momentary characteristics of the network and on the duration of the observation interval. An alarm is triggered if the following condition is violated:

$$NVR(t) < \varepsilon \; \forall \; t$$

The condition states that a node cannot change its neighbors from one point in time $t - \Delta t$ to another point in time t with a rate greater than or equal to ε. In Fig. 44 we depict an example of the usage of this threshold. The figure corresponds to a node that at a given point in time was attacked in our testbed. The attack was launched[2] for about one minute and it resulted in a flooding of packets announcing fake neighbors. We observe that during the attack, the NVR exceeded the threshold ε and therefore the attack was detected.

The use of the NVR for anomaly detection works better if the node has more than a certain number of neighbor nodes. If there are few neighbors the NVR will indicate high values even

2 Using the tool mdk3 [102] with these parameters: "[iface] s -f 2 -s 1000 [BSSID]".

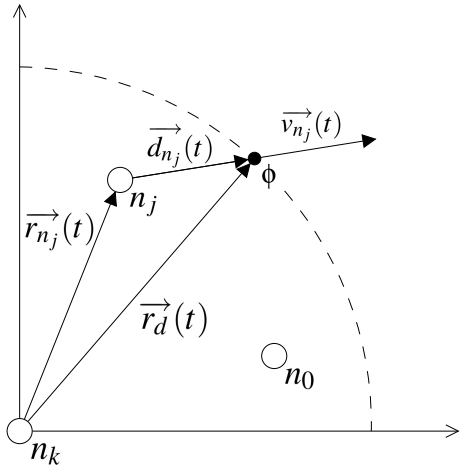


Figure 45: Analysis of the position and velocity of the neighbor nodes in order to estimate the link duration. The dashed line represents the transmission range of n_k cropped to the depicted quadrant.

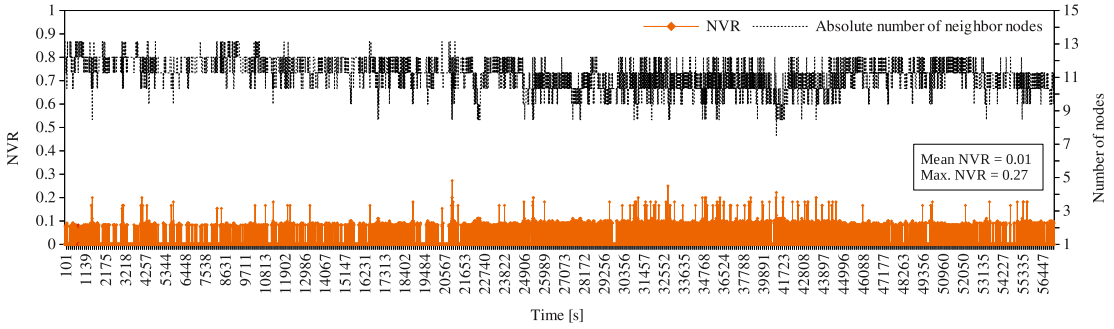


Figure 46: Neighbor Variation Rate of a node over 15 hours in a completely static wireless mesh network in our testbed. The observation interval is $\Delta t = 1s$.

under normal situations and it would be impractical to set a threshold level (i.e., if we only have one neighbor and it leaves the neighborhood, the NVR rises to 1.)

8.4.1 *Determining* ε

In order to determine the threshold ε we define a model to estimate the *link duration* of the neighbor nodes. There is a large body of work already done on link duration estimation and existing models can be applied to our work (e.g., [103–108]). As our goal is to apply our detection technique in a practical environment, we propose a simple approach that uses the speed of the neighbor nodes to estimate the link duration.

In practice, there are several possibilities for measuring the speed of a node. One indirect method is by estimating the speed based on the measured received signal strength. How to determine the speed and location is out of the scope of this chapter, but an approach can be found in [109].

We perform the analysis for a network containing i nodes. Let n_k be the node on which we analyze the NVR(t), and let n_0, n_1, ..., n_j ($j < i$) be neighbor nodes of n_k at a given time t. We analyze the node n_j. Let ϕ be a point located on the border of the disk delimiting the transmission range of n_k. Let $\overrightarrow{r_d}(t)$ be a position vector from n_k to ϕ. Let $\overrightarrow{v_{n_j}}(t)$ be the velocity of n_j. Let $\overrightarrow{r_{n_j}}(t)$ be a position vector from n_k to n_j. Let $\overrightarrow{d_{n_j}}(t)$ be the distance from n_j to the point ϕ:

$$\overrightarrow{d_{n_j}}(t) = \overrightarrow{r_d}(t) - \overrightarrow{r_{n_j}}(t)$$

Figure 45 depicts a representation of these parameters. Let $\tau_{n_j}(t)$ be the time needed for n_j to reach ϕ. Since we do not know the direction of $\overrightarrow{v_{n_j}}(t)$, for the sake of simplicity we assume it to be equal to the direction of $\overrightarrow{d_{n_j}}(t)$ (in a similar way, we can choose the location of the point $\phi : \angle\overrightarrow{r_d}(t) = \angle\overrightarrow{r_{n_j}}(t)$):

$$\tau_{n_j}(t) = \frac{|\overrightarrow{d_{n_j}}(t)|}{|\overrightarrow{v_{n_j}}(t)|}$$

Let $D(t)$ be the mean of the time needed for all the neighbor nodes of n_k to reach ϕ. In other words, $D(t)$ is the mean expected link duration:

$$D(t) \triangleq \frac{1}{j}\sum_{i=0}^{j} \tau_{n_i}(t) \tag{14}$$

As we want to find the time in which all the neighbor nodes of n_k have changed, this estimation should also consider the new nodes that come into the coverage area of n_k. Let us assume that the new nodes are uniformly distributed in the network and that they come into the transmission range of n_j with the same proportion as neighbors of n_j going out. We define the following variable:

$$\hat{D}(t) \triangleq \frac{D(t)}{2} \tag{15}$$

which expresses that after a period of time $\hat{D}(t)$ the neighbor nodes of n_k will have changed and therefore $NVR(\Delta t = \hat{D}(t)) \to \alpha$. $\alpha \in [0, 1]$ is the maximum value that $NVR(t)$ can reach. Ideally, if all the neighbors change then $\alpha = 1$. However, in practice the NVR does not yield 1 because of the mobility of nodes (nodes which do not move at all or which keep inside the neighborhood). See this parameter depicted in Fig. 43. We use the point $\Delta t = \hat{D}(t)$ to model $NVR(\Delta t)$ as an exponential function:

$$NVR(\Delta t) \triangleq \alpha - (1 - \beta)^{\frac{\Delta t}{\hat{D}(t)}} \tag{16}$$

$\beta \in [0, 1]$ is a tuning parameter. We have modeled the transmission range of a node as a disk. However, this is not the case in reality, where we have fading effects and different antenna radiation patterns. We have manually adjusted this parameter for the chapter and leave the development of an automatic mechanism to obtain it for future work.

Finally, we define the threshold ε as 50% bigger than $NVR(\Delta t), \forall \Delta t < \hat{D}(t)$. In our model we do not consider the behavior of the NVR for observation intervals bigger than $\hat{D}(t)$. In such cases, the neighbor nodes of n_k might escape the transmission range and come back again within the observation interval and no change to the variation of neighbors will be noticed (see Fig. 43). Ergo, the observation interval Δt should be adopted between $0 \leqslant \Delta t < \hat{D}(t)$. ε can be calculated at any time as:

$$\varepsilon(\Delta t) \triangleq 1.50 * NVR(\Delta t) \tag{17}$$

Example. Let us consider a network where the node n_k has a certain number of neighbor nodes at some point in time t. The average time within which all the neighbors have changed is $\hat{D}(t) = 60s$. α is set to 1 and β to 0.99. We calculate some values for the $NVR(\Delta t)$ and $\varepsilon(\Delta t)$, and show them in Table 14. For example, if we take $\Delta t = 5s$, we do not expect, according to our model, to have an NVR higher than $\varepsilon(\Delta t) = 0.48$ within that period (calculate from (16) and (17)). If it is higher, an alarm is triggered.

Table 14: Example of numerical calculations of $\varepsilon(\Delta t)$. We use in this example $\hat{D}(t) = 60s$.

Δt **[s]**	$NVR(\Delta t)$	$\varepsilon(\Delta t)$	**Comment**
1	0.07	0.10	Δt too short; normal fluctuations would be identified as attacks.
2	0.14	0.21	
5	0.32	0.48	System less tolerant to variations.
6.66	0.40	0.60	Suitable Δt (calculated from (18)).
10	0.54	0.80	System more tolerant to variations.
20	0.78	1	Δt too long; unpractical threshold.
30	0.90	1	
50	0.98	1	

8.4.2 *Special case: a static mesh network*

If the network is static or temporarily static, for example during the night hours, we do not expect changes in the node neighborhood. Ideally, $\hat{D}(t) \to \infty$ and $NVR \to 0$. However, in a real network, little fluctuations in the NVR occur due to unstable wireless links, failure on some nodes etc. For the completeness of our model, we define that the threshold $\varepsilon(\Delta t)$ should be set between $\varepsilon^{min} \leqslant \varepsilon(\Delta t) \leqslant 1$. An example of the behavior of the NVR in a static network is depicted in Fig. 46. The measurements were done for more than 15 hours in our testbed and the observation period was one second. The maximum NVR obtained was 0.27 and we can set $\varepsilon^{min} > 0.27$ for this case.

8.4.3 *Selection of the observation interval (Δt)*

The value of the observation interval should be selected by the system based on the following criteria: 1) it should be set higher than the value that yields $NVR(\Delta t) = \varepsilon^{min}$ so it can tolerate the normal fluctuations of the NVR in a real network; 2) it should be smaller than the value of $\hat{D}(t)$ that yields $NVR(\Delta t) = \alpha$. In addition, if we choose a value of Δt that results in an NVR close to α, the system will not have a large margin to discriminate normal from attack situations. In fact, if the value of Δt is too close to $\hat{D}(t)$ the threshold $\varepsilon(\Delta t)$ calculated in (17) would be close to/equal to 1 and any variation would be seen as a non-attacking situation. On the other hand, if the value of Δt yields the NVR to be too close to ε^{min}, the system will not be very tolerant to big changes of neighbors. A practical approach would be to select the value of Δt that yields $NVR(\Delta t) = 0.4$. We obtain from (16):

$$\Delta t = \hat{D}(t)\frac{\ln(\alpha - NVR)}{\ln(1-\beta)} \tag{18}$$

where NVR is replaced by 0.4. In the example of Table 14 the observation interval calculated from (18) is 6.66 s. Note that the threshold for this case is 0.60.

8.4.4 *What observation interval should be used when $\hat{D}(t)$ cannot be determined?*

This is especially important for the case of a static network (or a group of static nodes inside a MANET). In this particular case, we will have the condition of $\hat{D}(t) \to \infty$ and therefore the Δt cannot be calculated as in (18). Any observation interval would be valid in this situation. A

Table 15: Simulation parameters

Name	Value
Number of nodes	200
Scenario boundary	1000m x 1000m
Network protocol	IEEE 802.11
Mesh routing protocol	AODV
Initial position	Random
Mobility model	Gauss-Markov
Simulation duration	1800s (for Section 8.5.2), 3600s (for Section 8.5.2)

practical solution is to select the same observation interval employed for the mobile nodes of the mesh because these would influence the NVR of the static nodes when they move into their area.

8.4.5 *Periodical adjustment of the* Δt

The dynamic nature of the mesh network causes the topology to change over time. For example, during certain hours of the day, the number of mobile devices on the network can increase considerably, while at other times the network might be mostly static. Thus, it is very important for the detection system to periodically recalculate the value of Δt. As it depends on the time during which all the neighbor nodes have changed, this should also be estimated for the actual condition of the network (as determined in 8.4.1, this value is obtained from the average expected link duration $D(t)$ based on the measurement of the speed of the neighbor nodes). After Δt is recalculated, a new value of the threshold $\varepsilon(\Delta t)$ can be set and therefore the system can be adapted to the new environment.

8.5 EVALUATION

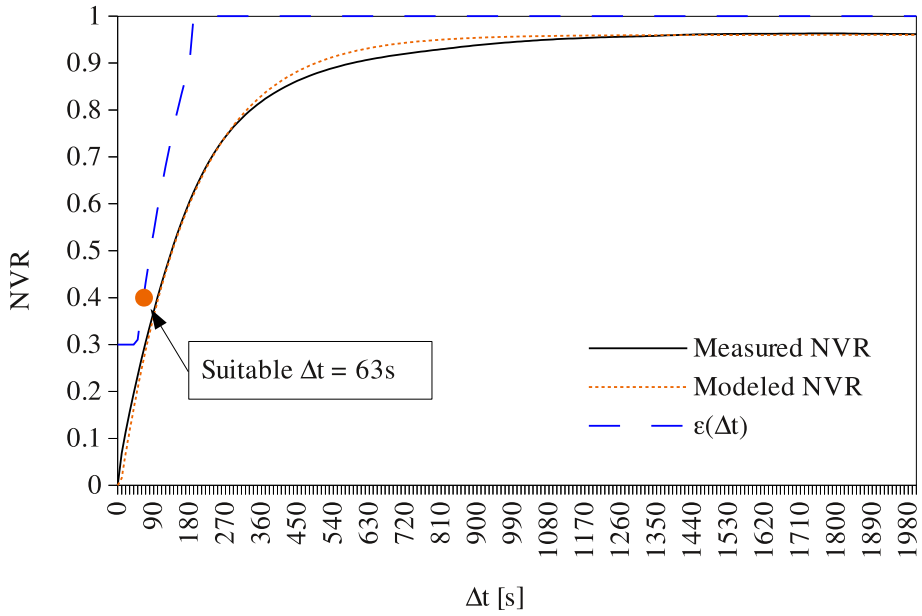
In this section we aim to validate our theoretical model and the detection technique defined in the previous sections. By means of extensive simulation, we perform the evaluation considering scenarios with a large number of nodes and high mobility. In addition, we implement and evaluate our model in a testbed running a wireless mesh network with mobile laptops and fixed nodes. We analyze and measure the NVR of an external dataset where several laptops were deployed in an outdoor environment.

8.5.1 *Simulation setup*

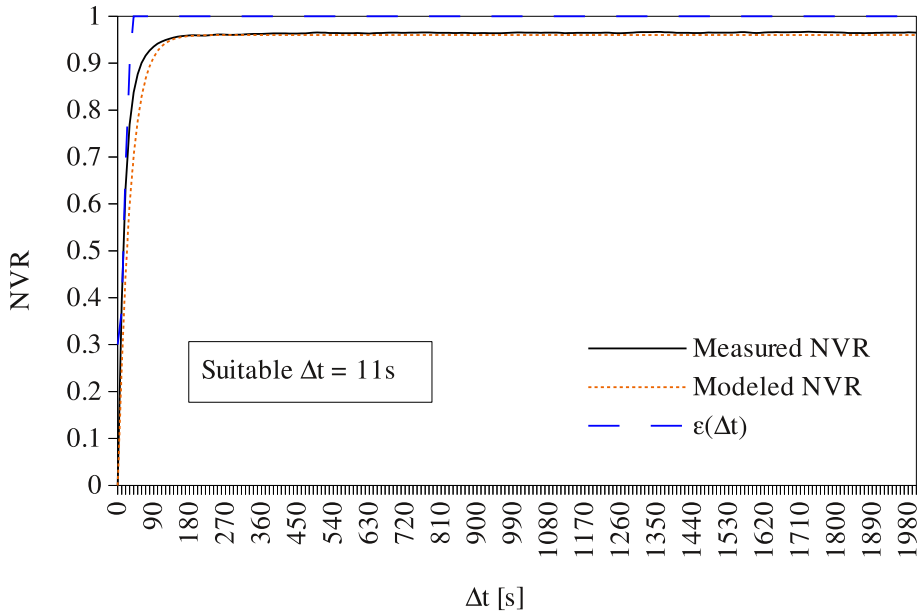
We performed the simulations using the software ns-3 [110]. The main parameters were adjusted as described in Table 15. We selected the Gauss-Markov [111] mobility model because it is temporal-dependent (the movement is affected by its history) and better fits the behavior of users in real life than other completely random models [112].

8.5.2 *Simulation results*

In this section we will show the results obtained from the simulation and the Neighbor Variation Rate calculated from the data gathered.



(a) Average speed of the nodes uniformly distributed in [0, 1] m/s.



(b) Average speed of the nodes uniformly distributed in [0, 10] m/s.

Figure 47: Accuracy of the modeled NVR compared to the measured NVR under two scenarios with different average speeds.

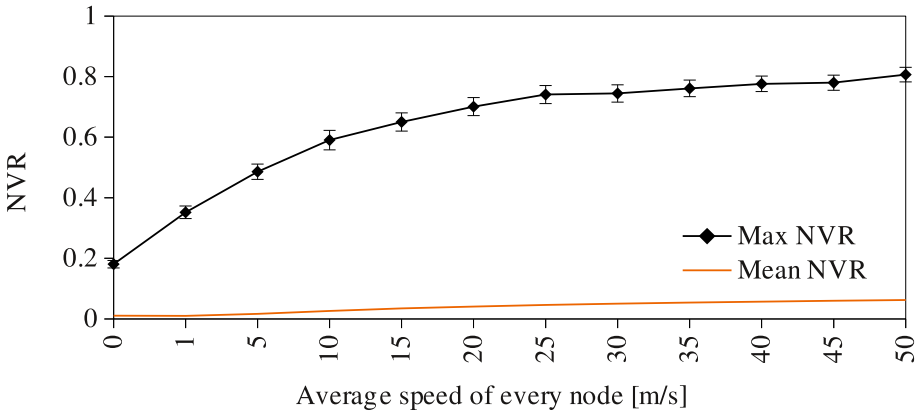


Figure 48: Mean and maximum NVR for different average speed of nodes (speed x uniformly distributed in [0, x] m/s).

NVR for different observation intervals

We evaluated the accuracy of the NVR modeled in (16) with the NVR measured in our simulations. For the model we used $\alpha = 0.96$ and $\beta = 0.96$ (coincidentally equal to each other). Figures 47a and 47b depict the curves of the modeled and measured NVR for two different scenarios. We performed 200 replications and the error bars are not shown because they are negligible. The expected time for the neighbors to leave the neighborhood ($\hat{D}(t)$) was measured from the simulations and introduced in (16) to obtain the modeled NVR. In addition, both figures depict the threshold $\varepsilon(\Delta t)$ calculated for every Δt from (17). For the examples, the ε^{min} is set to 0.3. In the first scenario, the average speed of the nodes is uniformly distributed in [0, 1] m/s. This represents a realistic network where the nodes move at a maximum speed similar to the speed of humans walking. We observe that a value of 63 s would be a suitable observation interval for this scenario as already detailed in Section 8.4.3. In the second scenario, the average speed is uniformly distributed in [0, 10] m/s. As the nodes move faster they change their neighbors faster than in the first scenario. Therefore, if we want to detect anomalies efficiently, we must select a lower observation interval. For this case, a suitable Δt calculated from (18) is 11 s.

NVR under different speeds

Figure 48 shows the plot of the mean and maximum NVR for different configurations of average speed of the nodes. The error bars for the curve "Max NVR" show the 95% confidence interval for 200 replications. The error bars for the curve "Mean NVR" are negligible on the scale of this figure. We have taken the lowest possible observation interval which is limited by the refreshment time of the neighbor nodes given by the routing protocol. In this case we used AODV and the refreshment time therefore is approximately 1 second. The minimal observation interval clearly imposes a limitation on the employment of the NVR for anomaly detection because at higher speeds, the maximum NVR is too high to set a threshold to identify anomalies. As we observe in Fig. 48, for speeds over 35 m/s (126 km/h) the mean NVR is relatively low but the NVR has peaks that rise over 0.6. Nevertheless, 35 m/s is a relatively high speed for multihop networks. If we consider networks formed by nodes moving at pedestrian speeds, the average speed should not exceed 1 m/s. Even in a city scenario, the cars would not be driving faster than 16 m/s. To employ our technique on highly mobile networks, i.e. networks with fast moving nodes, we should decrease the observation interval significantly as the routing protocols designed for such networks take faster moving nodes into account.

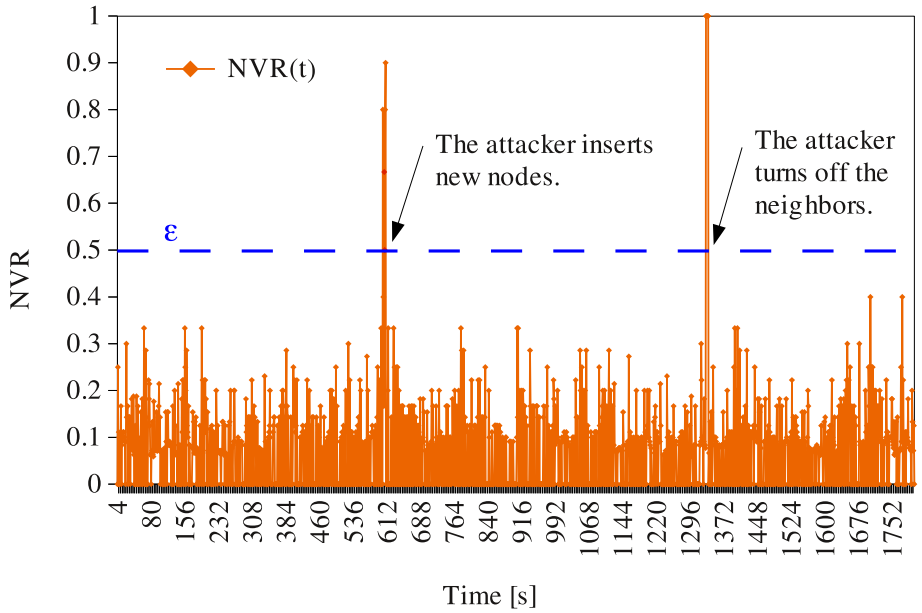


Figure 49: The first attack (left peak) introduces a big number of neighbor nodes (4 times). The second attack disconnects all the neighbors from the network (right peak).

NVR for anomaly/intrusion detection

In this experiment we show the feasibility of detecting anomalies by observation of the NVR as described in Section 8.4. However, we do not implement or analyze a complete intrusion detection system, but give insights on how our model would be applied to an IDS. In Fig. 49 we show the NVR of one of the nodes from the simulation. Two attacks are carried out. The first one introduces fake routing information provoking an increase of neighbor nodes in the routing table of the victim node. The second attack manages to power off the neighbors of the victim node. This results in the same effect as jamming as it suddenly interrupts all the links with neighbor nodes. For both attacks we can clearly observe a major increase in the NVR. The increment on the second attack is more abrupt because of the characteristics of the attack.

Due to the local nature of the mechanism and the usage of data already gathered by each node in the process of routing protocol setup, we do not expect any scalability issues when utilizing the NVR for intrusion detection. The NVR can be employed in wireless networks of all sizes and on systems of varying processing power.

8.5.3 *Testbed setup*

Hardware. We evaluate our technique in an indoor mesh network testbed composed of 18 static nodes installed on two floors of an office building and 5 mobile nodes. Each static node features an Alix 3D2 board [113] with a AMD Geode LX800 processor running Debian 5.0. The mobile nodes are Asus Eee-Pc 1005HA laptops running Ubuntu 11.04. The nodes run the open80211s [114] implementation of the IEEE 802.11s [27] protocol.

Environment and Mobility. We solicited five participants for our experimentation. Each participant carried one laptop and moved at pedestrian speed ($\approx$ 1 m/s). A representation of the layout of the testbed is depicted in Fig. 50. The trajectory of the participants carrying the laptops is also depicted. At the beginning of the experimentation all the participants were located at the point "C" marked in Fig. 50 and each of them took either the direction to "A" or to "B". The participants kept walking within the trajectory from "A" to "B" with random changes of direction and with short stops (10-30 s). We did several repetitions of 15 minutes each.

Figure 50: Experimental deployment of our proof-of-concept.

Measurement Methodology. We implemented Python and Bash scripts that determine the list of mesh peer links every second and store it in a CSV file. We ran them on each of the 23 nodes during the complete experimentation and analyzed the traces afterwards. During the experimentation, we launched the attacking tool mkd3 [102] version 7 (with the parameters: "[iface] s -f 2 -s 1000 [BSSID]") on one of the mobile nodes for 2 minutes. The attack sent packets to the network at a rate of 1000 packets/s announcing fake information (fake neighbors, fake routing information and fake data packets).

8.5.4 *Testbed results*

Attack detection

Figure 51 depicts the NVR of five nodes. The first four nodes are mobile nodes and the last one is static. The boxes represent the maximum values of NVR which occurred during the attack. The circles over the whiskers represent the maximum value of NVR without considering the attack. The dashed line crossing the plot horizontally represents the value of the threshold. The threshold was set to 0.6 for every node and we employ different observation intervals for each node (we calculated the suitable Δt from (18) for each node). We observe that the maximum NVR excluding the attack is below the defined threshold. For the mobile nodes these maximum values are higher than for the static node because of their mobility. The median NVR is considerably below these values (in every case in the range [0, 0.1]). The attack was detected in all cases.

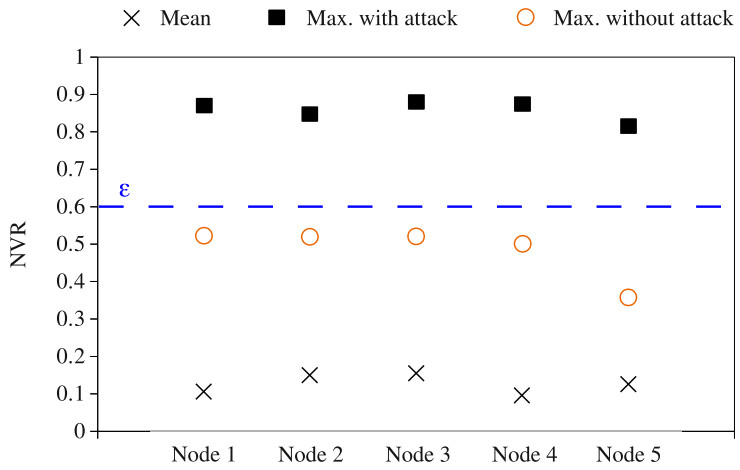


Figure 51: The attack was successfully detected. The first four nodes are mobile and the last one static.

The importance of correctly adjusting the observation interval (Δt)

In Fig. 52a we show the NVR of one of the fixed nodes in our testbed. In this case the NVR was calculated for an observation interval of one second. We note that when the node receives the effects of the attack, the NVR does not show a significant change. Even though it can be distinguished from the non-attack NVR, it might be hard to set a threshold level. This effect occurs when the observation interval is too short and the attacker inserts nodes gradually. The effect of the attack on the number of neighbor nodes is shown in Fig. 52c. The attack clearly raises the number of nodes from about 10 (real neighbors) to about 65 in a short period of time. However, the attack inserts them gradually and with a time separation longer than the observation interval selected (1 s). In Fig. 52c we can observe this phenomenon by looking at the points "a" to "d". The first effect of the attack is seen at the point "a" where the node advertises an increase of 20 neighbors. The NVR calculated at that point of time is not high. After some time the second effect of the attack is seen at the point "b". Again, the increase from 20 to less than 40 neighbors does not yield a high NVR. This reasoning holds for the points "c" and "d".

We now take a look at Fig. 52b which shows the NVR from the same node but for a larger observation interval of 30 seconds, and we can easily discriminate the non-attack from the attack situation. In this case, the longer observation interval ensures that the algorithm detects an increment of neighbor nodes from 10 to around 65. The observation interval should be carefully determined for every node and for every network condition as explained in Section 8.4.3.

Measurement of the NVR applied to an external dataset

In order to provide another realistic analysis of the NVR on a dynamic MANET, we study the dataset provided by [115]. The data was originally utilized to compare the performance of four different routing algorithms in an outdoor environment. The data employs 41 laptops with 802.11-enabled hardware moving randomly through a rectangular 225 m x 365 m athletic field. The routing algorithms employed were APRL, AODV, ODMRP, and STARA. The events of the routing algorithms were logged on each laptop together with the geographical position acquired from a GPS service running on each machine. As indicated by the authors in [115], some logging data is missing in the dataset, resulting in only 30 nodes for our study.

For the analysis in this chapter, we took advantage of the logs from the events of the AODV routing protocol generated on each laptop and we built a script to parse this data. Carefully examining the received HELLO packets at each node and considering their lifetime, we recreated the routing table of each node and measured the NVR using an observation interval of one second. We measured the NVR of 30 nodes and calculated its mean and maximum

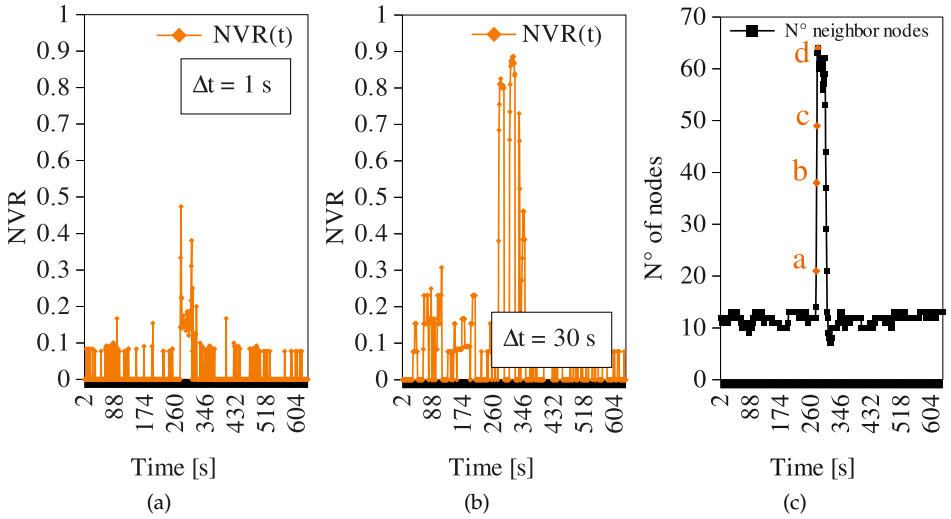
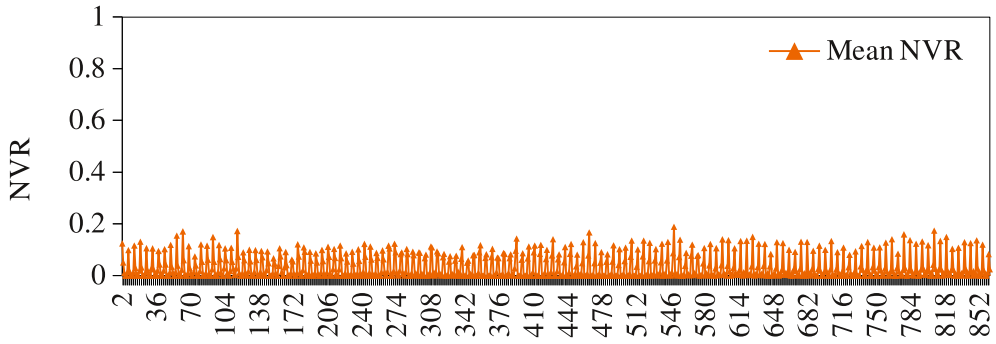
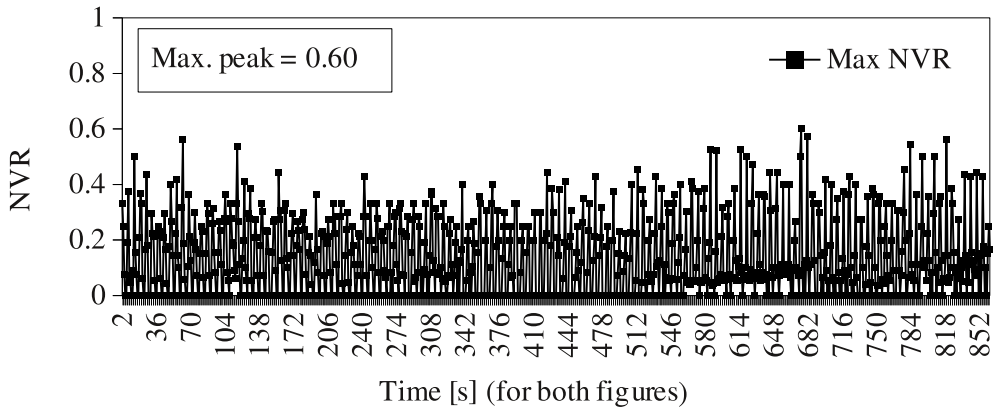


Figure 52: (a) and (b) show the NVR of a static node with different observation intervals. (c) depicts the absolute number of neighbors of the same node during the same time interval. The attack occurred between seconds 276 and 310.



(a) Mean of the NVR of 30 nodes.



(b) Maximum values of the NVR of the 30 nodes.

Figure 53: Analysis of the NVR calculated of the data provided by [115].

value. The results of this analysis are depicted in Figures 53a and 53b. By observing the mean we get a notion of how the NVR behaves in such a network. Due to the mobility, the mean is relatively high compared to the results obtained in Section 8.5.2. However, the mean by itself does not provide major insights (we show it for completeness). On the other hand, the observation of the maximum values is of special interest because we can analyze whether it is possible to define a threshold for anomaly detection or not. In this particular case, the maximum NVR got a peak of 0.60 because of the mobility, but even with this peak we can still define a threshold for the NVR, for example $\varepsilon = 0.75$.

8.6 CONCLUSION AND FUTURE WORK

We have shown that a lightweight metric can be useful for anomaly detection, especially in wireless multihop networks. These networks are often composed of resource constrained nodes where the employment of techniques that require high CPU or memory usage is not possible. By analyzing the rate of change of the neighbor nodes (one-hop distance), we defined a new metric and delimited a threshold to detect anomalies according to the mobility status of the network. We have shown that our technique can be easily deployed in real-world networks and that it is efficient for anomaly detection.

Moreover, our detection model is generic and can be integrated into any current IDS and we plan to do so in the future. Also further analysis on the detection accuracy needs to be performed when implementing the NVR as a detection mechanism on an IDS. In addition, we hope that with these contributions we can motivate researchers and developers to make use of our approach in current deployments of intrusion detection systems and gather further insights on large-scale wireless multihop networks.

9 CONCLUSIONS

> The user's going to pick dancing pigs over security every time.
>
> Bruce Schneier

9.1 SUMMARY AND CONCLUSIONS

Our fundamental research question of this dissertation was if there is a practical and efficient design alternative to the distributed deployment of intrusion detection sensors in wireless multihop networks, assuming resource-constrained network nodes and network protocols that cannot easily be modified. We have answered this question positively. We have introduced an innovative active-probing technique for intrusion detection in WMNs, we have investigated design elements and trade-offs, and we have performed substantial experiments to demonstrate that the technique is indeed practical and efficient.

We have presented our core solution in four main blocks, as depicted in Fig. 1 of the introductory chapter. The first block (Chapters 1 to 4) introduced the problem and design of the solution. In a review of the literature, we found that distributed intrusion detection is not actually practical for wireless multihop networks. As a result, we proposed an active-probing technique and presented its design. During the design phase, we performed our very first experiments, which presented a couple of practical challenges. These challenges brought us to our second block (Chapter 5), the study of the parameters of the system. Since there are no existing active techniques for intrusion detection in wireless multihop networks, to the best of our knowledge, we had to first identify the most critical parameters of active probing: For example, how many testing packets to send, how long time intervals between them are, how the link quality affects detection, among others. And secondly, we had to perform experiments and understand how to adjust them correctly. We identified another big challenge in active probing, namely how to reduce the number of probes that the intrusion detection has to transmit. This challenge led us to our third block (Chapter 6). Using statistics we designed a solution that is able to minimize the number of probes to transmit and select the probes on the fly that better fit the actual inquiries. The proposed solution is a general mathematical model that can also be applied in other systems where decisions on selections have to be made. In the fourth block, we finally focused on intrusion detection per se (Chapter 7). With an understanding of the essential parameters and the inference module designed and tested, we implemented an active-probing-based network intrusion detection system in an indoor office testbed based on 16 mesh fixed nodes and 2 mobile mesh nodes. We used available security tools and also designed our own in order to implement different real attacks with different deployment and detection complexities. The attack detection has been successful, we have been able to detect the attacks with over 90% detection rate. In addition, we demonstrated that our implementation is lightweight in network overhead. We measured that the traffic generated by our proof of concept was under 6% of the total traffic when there no background traffic was present. With background traffic, the generated traffic by the IDS was in some cases less than 0.1%. These are important results because we can argue that without introducing unnecessary overhead on the network nodes, as in distributed intrusion detection systems, our active-probing IDS can detect attacks with a minimal usage of the network traffic. We have made our implementation open source and freely accessible, including documentation.

9.2 OUTLOOK

We have introduced a new concept on intrusion detection for wireless multihop network and performed an in-depth study of its fundaments. Active-probing intrusion detection is a complex field and there are of course future challenges to address. Some of these include:

- *Heterogeneous security environment.* An interesting research direction is the practical usage of active-probing-based intrusion detection systems in complex environments where other security systems coexist together. Such an environment can be composed of security policies, security requirements and security components. Firewalls and URL filters might help to detect or mitigate certain attacks, but they might not be sufficient for others. We do not believe that active can completely replace passive, but active and passive intrusion detection and prevention systems can be deployed with different tasks or objectives assigned. An interesting question is how such a complex environment can be optimized to fulfill all the requirements while keeping deployment costs minimal?
- *Mobility of the detection node.* We have presented and extended the mobility of the active-probing IDS to a certain extent. However, research on mobile networks is an interesting area for future work on active probing. An active-probing-based IDS can be deployed on robots in disaster areas, for example, and check that the underlying mesh infrastructure works. Or in vehicular networks, where not only the network security but also the user's safety can be compromised in the presence of attackers.

In summary, we introduced active probing for security and demonstrated that it is practical for intrusion detection in wireless multihop communications. We believe that with the proliferation of wireless multihop networks for their very vast usages, the need for intrusion detection will increase and active-probing-based systems are going to be used both in practical deployment as well as in research. We hope that our theoretical work and our implementation, DogoIDS, will motivate colleagues to continue exploring this field.

[1] G. Vasilakis, G. Perantinos, I. G. Askoxylakis, N. Mechin, V. Spitadakis, and A. Traganitis. Business opportunities and considerations on wireless mesh networks. In *Proceedings of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM'09)*. IEEE, 2009.

[2] T. B. Lee. Multi-hop matters: the state of wireless mesh networking. Online, 2009. `http://goo.gl/KdfCQ`, last accessed on October 1st, 2014.

[3] SAIL Project - D.A.7 New business models and business dynamics of the future networks. Online, 2013. `http://www.sail-project.eu`, last accessed on October 1st, 2014.

[4] J. Rebello. More than 1 billion devices to have embedded wireless networking capability. Online, 2011. `http://goo.gl/XKknr`, last accessed on October 1st, 2014.

[5] The Serval Project. Online, 2014. `http://www.servalproject.org`, last accessed on October 1st, 2014.

[6] MeshScape(R) wireless sensor networking system. Online, 2012. `http://goo.gl/7JsaN`, last accessed on October 1st, 2014.

[7] CAR 2 CAR Communication Consortium. Online, 2014. `http://goo.gl/t44tR`, last accessed on October 1st, 2014.

[8] Daintree Networks, Applying Mesh Networking to Wireless Lighting Control. Online, 2014. `http://goo.gl/lkflF`, last accessed on October 1st, 2014.

[9] Freifunk. Online, 2014. `http://wiki.freifunk.net`, last accessed on October 1st, 2014.

[10] Thread Group. Online, 2014. `http://www.threadgroup.org`, last accessed on October 1st, 2014.

[11] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *Springer-Verlag Wireless Networks*, 9(5), Sep 2003.

[12] A. Mishra, K. Nadkarni, and A. Patcha. Intrusion detection in wireless ad hoc networks. *IEEE Wireless Communications*, Feb 2004.

[13] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'00)*. ACM, 2000.

[14] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN'03)*. ACM, 2003.

[15] C. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. N. Levitt. A specification-based intrusion detection system for AODV. In *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03)*. ACM, 2003.

[16] F. Hugelshofer, P. Smith, D. Hutchison, and N. J. P. Race. OpenLIDS: A lightweight intrusion detection system for wireless mesh networks. In *Proceedings of the 15th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'09)*. ACM, 2009.

[17] G. Vigna, S. Gwalani, K. Srinivasan, E. M. Belding-Royer, and R. A. Kemmerer. An intrusion detection tool for AODV-based ad hoc wireless networks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*. IEEE, 2004.

[18] R. V. Boppana and X. Su. On the effectiveness of monitoring for intrusion detection in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, Aug 2011.

[19] H. Yih-Chun and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, May-Jun 2004.

[20] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, and A. Jamalipour. A survey of routing attacks in mobile ad hoc networks. *IEEE Wireless Communications*, Oct 2007.

[21] F. Oliviero and S. P. Romano. A reputation-based metric for secure routing in wireless mesh networks. In *Proceedings of the IEEE Global Communications Conference 2008 (GLOBECOM'08)*. IEEE, 2008.

[22] I. Poole. Self organising networks. Online, 2014. `http://goo.gl/JFvTiM`, last accessed on October 1st, 2014.

[23] A. K. Pathan. *Security of self-organizing networks: MANET, WSN, WMN, VANET*. CRC Press, Hoboken, 2010.

[24] T. Braun, A. Kassler, M. Kihl, V. Rakocevic, V. Siris, and G. Heijenk. Multihop wireless networks. In V. Siris, T. Braun, F. Barcelo-Arroyo, D. Staehle, G. Giambene, and Y. Koucheryavy, editors, *Traffic and QoS Management in Wireless Multimedia Networks*, volume 31 of *Lecture Notes in Electrical Engineering*. Springer US, 2009.

[25] I.F. Akyildiz and W. Xudong. A survey on wireless mesh networks. *IEEE Communications Magazine*, 43(9), Sep 2005.

[26] IEEE standard 802 part 11. Online, 2007. `http://standards.ieee.org/about/get/802/802.11.html`, last accessed on October 1st, 2014.

[27] IEEE standard 802 part 11 amendment 10: mesh networking. Online, 2011. `http://standards.ieee.org/findstds/standard/802.11s-2011.html`, last accessed on October 1st, 2014.

[28] G. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke. IEEE 802.11s: The WLAN mesh standard. *IEEE Wireless Communications*, Feb 2010.

[29] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561 Experimental, IETF. Online, 2003. `http://rfc.net/rfc3561.txt`.

[30] R. C. Carrano, L. C. S. Magalhaes, D. C. M. Saade, and C. V. N. Albuquerque. IEEE 802.11s multihop MAC: A tutorial. *IEEE Communications Surveys Tutorials*, First quarter 2011.

[31] Z. Zhao, H. Hu, G.-J. Ahn, and R. Wu. Risk-aware mitigation for MANET routing attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(2), 2012.

[32] P. Goyal, V. Parmar, , and R. Rishi. MANET: vulnerabilities, challenges, attacks, applications. *International Journal of Computational Engineering & Management*, 11, Jan 2011.

[33] M. Abdelhaq. Detecting sleep deprivation attack over MANET using a danger theory-based algorithm. *International Journal of New Computer Architectures and their Applications (IJNCAA)*, 3(1), 2011.

[34] S. Roy, V.G. Addada, S. Setia, and S. Jajodia. Securing MAODV: attacks and countermeasures. In *Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05)*. IEEE, 2005.

[35] S. Dokurer, Y.M. Erten, and C.E. Acar. Performance analysis of ad-hoc networks under black hole attacks. In *Proceedings of the 2007 annual IEEE SoutheastCon conference*. IEEE, 2007.

[36] S. Buchegger, C. Tissieres, and J.-Y. Le Boudec. A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do? In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*. IEEE, 2004.

[37] I. Aad, J.-P. Hubaux, and E.W. Knightly. Impact of denial of service attacks on ad hoc networks. *IEEE/ACM Transactions on Networking*, 16(4), 2008.

[38] J. von Mulert, I. Welch, and W.K.G. Seah. Security threats and solutions in MANETs: A case study using AODV and SAODV. *Elsevier Journal of Network and Computer Applications*, 35(4), 2012.

[39] F. Kandah, Y. Singh, and C. Wang. Colluding injected attack in mobile ad-hoc networks. In *IEEE INFOCOM WKSHPS Machine-to-Machine Communications and Networking (M2MCN)*, 2011.

[40] K. Graffi, P.S. Mogre, M. Hollick, and R. Steinmetz. Detection of colluding misbehaving nodes in mobile ad hoc and wireless mesh networks. In *Proceedings of the 2007 Global Telecommunications Conference (GLOBECOM '07)*. IEEE, 2007.

[41] H. Weerasinghe and H. Fu. Preventing cooperative black hole attacks in mobile ad hoc networks: simulation implementation and evaluation. In *Proceedings of the 2007 International Conference on Future Generation Communication and Networking (FGCN'07)*. IEEE, 2007.

[42] A. Mishra, R. Jaiswal, and S. Sharma. A novel approach for detecting and eliminating cooperative black hole attack using advanced DRI table in Ad hoc Network. In *Proceedings of the IEEE 3rd International Advance Computing Conference (IACC'13)*. IEEE, 2013.

[43] Y.-A. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. In *Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004.

[44] X. Wu and D.K.Y. Yau. Mitigating denial-of-service attacks in MANET by incentive-based packet filtering: A game-theoretic approach. In *Proceedings of the 3rd IEEE International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm'07)*. IEEE, 2007.

[45] S.M. Glass, V. Muthukkumarasamy, and M. Portmann. Detecting man-in-the-middle and wormhole attacks in wireless mesh networks. In *Proceedings of the 2009 IEEE International Conference on Advanced Information Networking and Applications (AINA'09)*. IEEE, 2009.

[46] E.Y. Vasserman and N. Hopper. Vampire attacks: draining life from wireless ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 12(2), 2013.

[47] N. Alsharif, A. Wasef, and X. Shen. Mitigating the effects of position-based routing attacks in vehicular ad hoc networks. In *Proceedings of the 2011 IEEE International Conference on Communications (ICC'11)*. IEEE, 2011.

[48] P.S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz. A security framework for wireless mesh networks. *John Wiley & Sons Wireless Communications and Mobile Computing Special Issue: Architectures and Protocols for Wireless Mesh, Ad Hoc, and Sensor Networks*, May 2010.

[49] T. Shu and M. Krunz. Detection of malicious packet dropping in wireless ad hoc networks based on privacy-preserving public auditing. In *Proceedings of the 5th ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec'12)*. ACM, 2012.

[50] D. Djenouri, O. Mahmoudi, M. Bouamama, D. Llewellyn-Jones, and M. Merabti. On securing MANET routing protocol against control packet dropping. In *Proceedings of the IEEE International Conference on Pervasive Services*. IEEE, 2007.

[51] M.S. Alkatheiri, J. Liu, and A.R. Sangi. AODV routing protocol under several routing attacks in MANETs. In *Proceedings of the IEEE 13th International Conference on Communication Technology (ICCT'11)*. IEEE, 2011.

[52] Y.C. Hu, A. Perrig, and D.B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2), 2006.

[53] M. Khabbazian, H. Mercier, and V.K. Bhargava. Wormhole attack in wireless ad hoc networks: analysis and countermeasure. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'06)*. IEEE, 2006.

[54] E.M. Shakshuki, N. Kang, and T.R. Sheltami. EAACK—a secure intrusion-detection system for MANETs. *IEEE Transactions on Industrial Electronics*, 60(3), Mar 2013.

[55] C. Tseng, S. Wang, C. Ko, and K. Levitt. DEMEM: distributed evidence-driven message exchange intrusion detection model for MANET. In D. Zamboni and C. Kruegel, editors, *Recent Advances in Intrusion Detection*, volume 4219 of *Lecture Notes in Computer Science*. Springer Berlin-Heidelberg, 2006.

[56] L. Mechtri, F.D. Tolba, and S. Ghanemi. MASID: multi-agent system for intrusion detection in MANET. In *Proceedings of the 2012 9th IEEE International Conference on Information Technology: New Generations (ITNG'12)*, Apr 2012.

[57] O. Kachirski and R. Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *Proceedings of the 36th IEEE Annual Hawaii International Conference on System Sciences (HICSS'03)*. IEEE, 2003.

[58] R. Shrestha, K.H. Han, D.Y. Choi, and S.J. Han. A novel cross layer intrusion detection system in MANET. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'10)*, April 2010.

[59] S. Şen and J. A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *Proceedings of the 2nd ACM conference on Wireless Network Security (WiSec'09)*. ACM, 2009.

[60] F. Anjum and R. Talpade. LiPaD: lightweight packet drop detection for ad hoc networks. In *Proceedings of the 60th IEEE Vehicular Technology Conference (VTC Fall 2004)*. IEEE, Sep 2004.

[61] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C.Y. Tseng, and T. Bowen. A general cooperative intrusion detection architecture for MANETs. In *Proceedings of the 3rd IEEE International Workshop on Information Assurance*, March 2005.

[62] A. Nadeem and M. Howarth. Adaptive intrusion detection & prevention of denial of service attacks in MANETs. In *Proceedings of the 2009 ACM International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC'09, New York, NY, USA, 2009. ACM.

[63] G.F. Cretu, J.J. Parekh, K. Wang, and S.J. Stolfo. Intrusion and anomaly detection model exchange for mobile ad-hoc networks. In *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, Jan 2006.

[64] A. Mitrokotsa, N. Komninos, and C. Douligeris. Intrusion detection with neural networks and watermarking techniques for MANET. In *Proceedings of the IEEE International Conference on Pervasive Services*, Jul 2007.

[65] J.M. Orset, B. Alcalde, and A. Cavalli. An EFSM-based intrusion detection system for ad hoc networks. In *Proceedings of the 3rd International Conference on Automated Technology for Verification and Analysis (ATVA'05)*, Berlin, Heidelberg, 2005. Springer-Verlag.

[66] A. Nadeem and M.P. Howarth. A survey of MANET intrusion detection amp; prevention approaches for network layer attacks. *IEEE Communications Surveys Tutorials*, 15(4), Apr 2013.

[67] J. Cordasco and S. Wetzel. An attacker model for MANET routing security. In *Proceedings of the 2nd ACM conference on Wireless Network Security (WiSec'09)*. ACM, 2009.

[68] R. do Carmo and M. Hollick. DogoIDS: a mobile and active intrusion detection system for IEEE 802.11s wireless mesh networks. In *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec'13)*. ACM, 2013.

[69] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Elsevier Computer Networks*, 31(9), April 1999.

[70] F. Rochat, P. Schöneich, B. Lüthi, H. Bleuler, R. Moser, and F. Mondada. Cy-mag3D: a simple and miniature climbing robot with advance mobility in ferromagnetic environment. *Industrial Robot: An International Journal, Emerald Group*, 2011.

[71] G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, P. Schöneich, F. Rochat, F. Mondada, and R. Moser. Highly compact robots for inspection of power plants. In *Proceedings of the 1st International Conference on Applied Robotics for the Power Industry (CARPI'10)*. IEEE, 2010.

[72] W. Baetz, A. Kroll, and S. Soldan. On gas leak detection of pressurised components by using thermograms and pattern recognition algorithms. In *Proceedings of the 8th International Conference on NDE in Relation to Structural Integrity for Nuclear and Pressurised Components*. European Commission, Joint Research Center, Institute for Energy, 2010.

[73] M. Muuse. The story of the PING program. Online, 1999. `http://www.webcitation.org/5saCKBpgH`, last accessed on October 1st, 2014.

[74] M. Portoles-Comeras, J. Mangues-Bafalluy, and M. Requena-Esteso. Multi-radio based active and passive wireless network measurements. In *Proceedings of the 2006 4th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, April 2006.

[75] L. Ciavattone, A. Morton, and G. Ramachandran. Standardized active measurements on a tier 1 IP backbone. *IEEE Communications Magazine*, 41(6), June 2003.

[76] W.W.T. Fok, X. Luo, R. Mok, W. Li, Y. Liu, E.W.W. Chan, and R.K.C. Chang. MonoScope: Automating network faults diagnosis based on active measurements. In *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM'13)*, May 2013.

[77] M. Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, New York, NY, USA, 2010. ACM.

[78] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6), Aug 2003.

[79] J. Liu and B. Chen. Internet traffic characterization based on active network measurement. In *Proceedings of the 2010 6th IEEE International Conference on Wireless Communications Networking and Mobile Computing (WiCOM'10)*, Sep 2010.

[80] A. Barbuzzi, F. Ricciato, and G. Boggia. Discovering parameter setting in 3G networks via active measurements. *IEEE Communications Letters*, 12(10), Oct 2008.

[81] Réseaux IP Européens Network Coordination Centre (RIPE NCC) Atlas Project. Online, 2014. `https://atlas.ripe.net`, last accessed on October 1st, 2014.

[82] U. Shankar and V. Paxson. Active mapping: resisting nids evasion without altering traffic. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[83] R. Beverly, A. Berger, and G.G. Xie. Primitives for active internet topology mapping: toward high-frequency characterization. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, New York, NY, USA, 2010. ACM.

[84] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: degree-based vs. structural. *SIGCOMM Computer Communication Review*, 32(4), Aug 2002.

[85] IEEE Standard Radar Definitions. *IEEE Std 686-1997*, 1998.

[86] S.I. Ziegler. Positron emission tomography: principles, technology, and recent developments. *Elsevier Nuclear Physics A*, 752, Apr 2005.

[87] R. do Carmo and M. Hollick. Analyzing active probing for practical intrusion detection in wireless multihop networks. In *Proceedings of the 11th IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (WONS'14)*. IEEE/IFIP, 2014.

[88] J. Sztrik. Basic queuing theory, lecture note. Online, 2012. `http://goo.gl/0JzYGp`, last accessed on October 1st, 2014.

[89] E. Altman and A. Jean-Marie. The loss process of messages in an M/M/1/K queue. In *Proceedings of the 13th IEEE Networking for Global Communications (INFOCOM'94)*. IEEE, 1994.

[90] O. Ait-Hellal, E. Altman, A. Jean-Marie, and I.A. Kurkova. On loss probabilities in presence of redundant packets and several traffic sources. *Elsevier Performance evaluation*, 36-37, 1999.

[91] I. Rish, M. Brodie, N. Odintsova, M. Sheng, and G. Grabarnik. Real-time problem determination in distributed systems using active probing. In *IEEE/IFIP Network Operations and Management Symposium (NOMS'04)*. IEEE, 2004.

[92] M. Brodie, I. Rish, and S. Ma. Intelligent probing: a cost-effective approach to fault diagnosis in computer networks. *IBM Syst. J.*, 41(3), July 2002.

[93] R. do Carmo, J. Hoffmann, V. Willert, and M. Hollick. Making active-probing-based network intrusion detection in wireless multihop networks practical: A bayesian inference approach to probe selection. In *Proceedings of the 39th IEEE Conference on Local Computer Networks (LCN '14)*. IEEE, 2014.

[94] C. M. Bishop. *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag New York, Inc., 2006.

[95] B. Das. Generating conditional probabilities for bayesian networks: easing the knowledge acquisition problem. *CoRR*, cs.AI/0411034, 2004.

[96] J.H. Cho, A. Swami, and I.R. Chen. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys Tutorials*, 13(4), Fourth Quarter 2011.

[97] R. do Carmo, M. Werner, and M. Hollick. Signs of a bad neighborhood: A lightweight metric for anomaly detection in mobile ad hoc networks. In *Proceedings of the 8th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet'12)*. ACM, 2012.

[98] J.P. Singh and P. Dutta. Temporal modeling of node mobility in mobile ad hoc network. *CIT*, 18(1), 2010.

[99] H.N. Nguyen and Y. Shinoda. A node's number of neighbors in wireless mobile ad hoc networks: a statistical view. In *Proceedings of the 8th IEEE International Conference on Networks (ICN'09)*, March 2009.

[100] I. Rhee, M. Shin, S. Hong, K. Lee, S.J. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM Transactions on Networking*, 19(3), June 2011.

[101] O.V. Drugan, T. Plagemann, and E. Munthe-Kaas. Non-intrusive neighbor prediction in sparse MANETs. In *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07)*, June 2007.

[102] mdk3. Online, 2009. `http://aspj.aircrack-ng.org/#mdk3`, last accessed on October 1st, 2014.

[103] Y.T. Wu, W. Liao, C.L. Tsao, and T.N. Lin. Impact of node mobility on link duration in multihop mobile networks. *IEEE Transactions on Vehicular Technology*, 58(5), June 2009.

[104] W. Viriyasitavat, F. Bai, and O.K. Tonguz. Dynamics of network connectivity in urban vehicular networks. *IEEE Journal on Selected Areas in Communications*, 29(3), March 2011.

[105] R.J. La and Y. Han. Distribution of path durations in mobile ad hoc networks and path selection. *IEEE/ACM Transactions on Networking*, 15(5), October 2007.

[106] F. Bai, N. Sadagopan, B. Krishnamachari, and A. Helmy. Modeling path duration distributions in MANETs and their impact on reactive routing protocols. *IEEE Journal on Selected Areas in Communications*, 22, 2004.

[107] M. Gerharz, C. de Waal, M. Frank, and P. Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, May 2002.

[108] C.L. Tsao, Y.T. Wu, W. Liao, and J.C. Kuo. Link duration of the random way point model in mobile ad hoc networks. In *Proceedings of the IEEE 2006 Wireless Communications and Networking Conference*, April 2006.

[109] H. Miura, K. Hirano, N. Matsuda, H. Taki, N. Abe, and S. Hori. Indoor localization for mobile node based on RSSI. In *Proceedings of the 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES'07)*. Springer-Verlag, September 2007.

[110] The Network Simulator – Version 3. Online, 2014. `http://www.nsnam.org`, last accessed on October 1st, 2014.

[111] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special Issue On Mobile Ad Hoc Networking: Research, Trends And Applications*, 2, 2002.

[112] F. Bai and A. Helmy. A survey of mobility models in wireless adhoc networks. In A. Safwat, editor, *Wireless Ad Hoc and Sensor Networks*. Springer Verlag, Berlin, 2007.

[113] Alix board 3D2 wiki. Online, 2009. http://goo.gl/fTzh0, last accessed on October 1st, 2014.

[114] open80211s project. Online, 2014. http://open80211s.org, last accessed on October 1st, 2014.

[115] R.S. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan. CRAWDAD data set dartmouth/outdoor (v. 2006-11-06). Online, November 2006. http://crawdad.cs.dartmouth.edu/dartmouth/outdoor, last accessed on October 1st, 2014.

LIST OF ACRONYMS

AODV	Ad hoc On-Demand Distance Vector
AMPE	Authenticated Mesh Peering Exchange
ABID	Anomaly-Based Intrusion Detection
APE	Active-Probing Engine
AP-NIDS	Active-Probing-Based Network Intrusion Detection System
BSS	Basic Service Set
CPT	Conditional Probability Table
DS	Distribution System
FPR	False Positive Rate
HWMP	Hybrid Wireless Mesh Protocol
IDS	Intrusion Detection System
IE	Inference Engine
KBID	Knowledge-Based Intrusion Detection
MANET	Wireless Mobile Ad hoc Network
MBSS	Mesh Basic Service Set
MPM	Mesh Peering Management Protocol
MSTA	Mesh Station
NVR	Neighbor Variation Rate
NIDS	Network Intrusion Detection System
PERR	Path Error
PREP	Path Response
PREQ	Path Request
RSSI	Received Signal Strength Indicator
SAE	Simultaneous Authentication of Equals
SON	Self-Organizing Network
SBID	Specification-Based Intrusion Detection
TCP	Transmission Control Protocol
TTL	Time To Live
VANET	Vehicular Ad hoc Network
WMN	Wireless Multihop Network
WSN	Wireless Sensor Network

A APPENDIX

A.1 GENERAL SETUP FOR DIFFERENT EXPERIMENTS

If not otherwise described, we follow a general setup for the experiments of this dissertation. We employ an indoor testbed composed of 16 mesh nodes installed on two floors of an office building. The important settings of our experiments are presented in Table 16. The distribution of the nodes of our testbed is shown in Fig. 54. The figure also shows a snapshot of the peer links created between the nodes. The AP-NIDS represents the deployment of our proof-of-concept DogoIDS. We use two netbooks for intrusion detection system purposes. One netbook runs DogoIDS, while the other only acts as a well-known final destination for the testing packets generated by DogoIDS. Note that we use two netbooks only to emulate a device with 2 network interfaces.

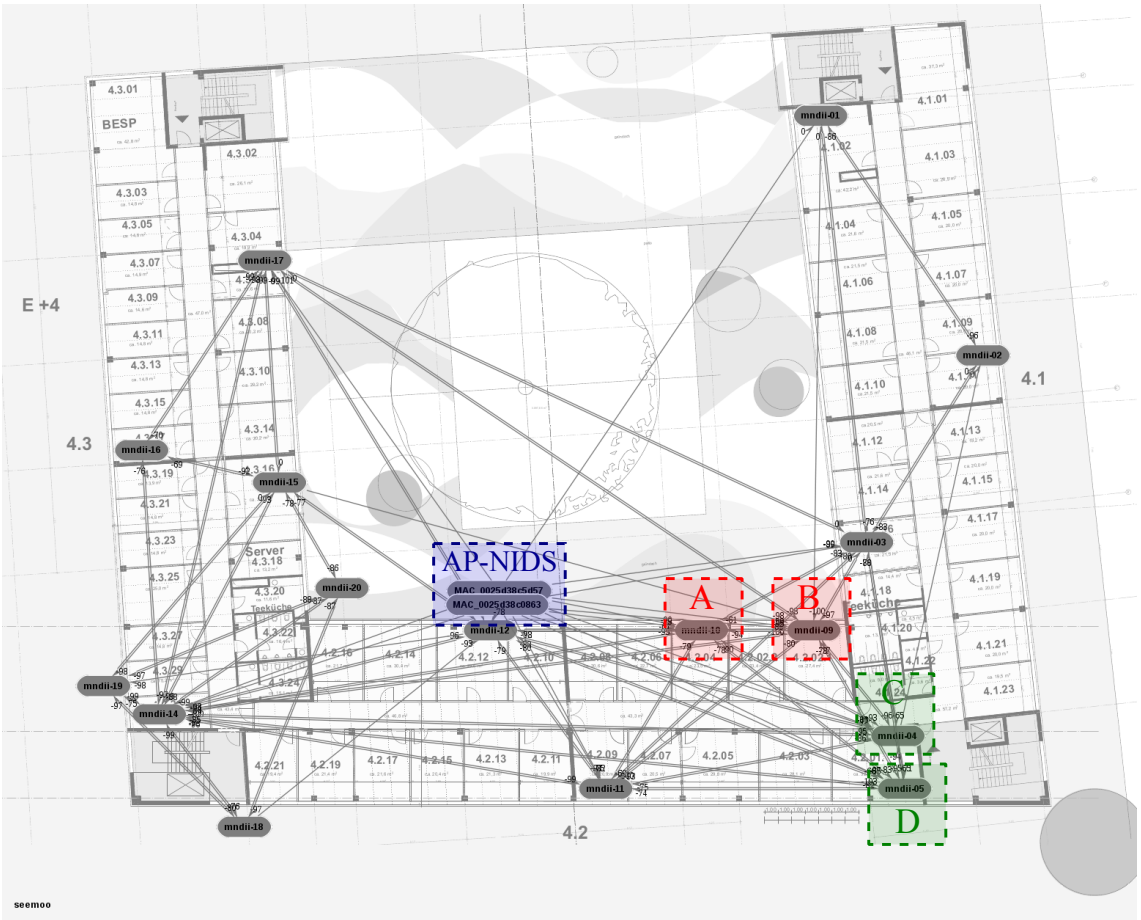


Figure 54: Setup of our mesh testbed and a snapshot of the network topology. The location of the AP-NIDS nodes does not represent their exact physical location (they are located inside the room left of node A).

A.1.1 *Background traffic*

We set 25 Mbit/s as the maximum achievable throughput between two nodes one-hop distanced, and we employ different percentages of it to vary the load. We refer to 0 Mbit/s when we do not generate extra background traffic.

A.1.2 *Replications*

In Table 16, the "Number of analyses" is the number of complete active-probing analyses that DogoIDS performs per replication. In other words, it represents the number of samples taken. We take a value large enough to be statistically significant. We perform 3 replications per experiment. Note that we study one variable per experiment. For example, for the experimentation of "Number of repetitions of testing packets", we perform one experiment per background traffic scenario and per number of repetitions transmitted (6 different repetitions of testing packets * 5 different background traffic values = 30 experiments). If we do 50 analyses per experiment, 3 times replicated, at the end we collect 30*50*3=1500 samples. Unless stated otherwise, the error bars in the figures of this dissertation represent the standard deviation.

A.2 EXPERIMENTS OF CHAPTER 5

We depict the configuration for the different experiments of this chapter in Tables 17, 18, 19, and 20.

A.3 EXPERIMENTS OF CHAPTER 6

The setup for the evaluation of the Bayes classifier with selective dropping of packets without background traffic is depicted in Table 21. The evaluation of the Bayes classifier with selective dropping of SSH and UDP packets with varying background traffic is depicted in Table 22.

A.3.1 *Setting up the attacks*

We decided to evaluate the experiments in this chapter by focusing on the *selective dropping of packets*. As explained in [49], this attack can severely degrade the performance of the network if the small number of highly critical control packets are dropped. In addition, it allows several rules to be defined for different services. We implemented the attacks of Table 6 by modifying the module `mac80211` of the wireless-testing Linux kernel. In particular, we modified some of the functions of the file *net/mac80211/rx.c* in order to allow the dropping of packets according to their transport protocol (TCP and/or UDP) and port number.

Table 16: General parameters for all experiments

Parameter	Value/Description
	Mesh network description
Number of nodes turned on	16 fixed nodes (nodes 6,7,8,13 out of service) + 2 netbooks
Fixed/Netbooks node system	`wireless-testing` kernel 3.5.0-rc4 with driver `ath5k` (fixed nodes) and `ath9k` (netbooks) from `backports` package 3.11-rc3-1 compiled with default options (`make defconfig-ath5k`)
TX power	20 dBm
Data rate	Auto (Minstrel algorithm)
RTS/CTS	Off
	IDS setup
IDS node	2 netbooks (for emulating 2 interfaces)
Target node	Node B (DogoIDS only analyzes this node)
Average TX bitrate for target node	48 Mbit/s
Average signal level to target node	-57 dBm
Capture time after the transmission	2 s
	Background traffic
Traffic tool	`iperf`
Traffic type	UDP packets, CBR
Server node	Node B
Server command	`iperf -s -u -i 1`
Client node	Node A
Client command	`iperf -u -c 192.168.0.10 -b 6.25M -t 1000 -i 1` for generating 6.25 Mbit/s
	Data collection
IDS metrics	FPR, detection time, throughput, CPU usage, memory load
IDS technique	Log files from DogoIDS, data from `ps`
Target node metrics	Dropping rate, throughput, debugging information from module `ieee80211`
Target node technique	`tcpdump` and every 1 s all information from `/sys/kernel/debug/ieee80211` is saved in a separate directory
	Experimentation
Number of analyses (DogoIDS)	50 per variable studied
Replications	3

Table 17: Parameters for experiment: Number of repetitions of testing packets (1 PREQ per Analysis)

Parameter	Value/Description
	IDS setup
Rules active	Dropping of UDP
Repetitions of testing packets	1, 2, 3, 4, 5, 10
Interval between replications	500 ms
Other information	One PREQ packet is transmitted to the target node at the beginning of each analysis
Packets transmitted per analysis	2 (1 PREQ), 3 (1 PREQ), 4 (1 PREQ), 5 (1 PREQ), 6 (1 PREQ), 11 (1 PREQ)
	Background traffic
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s

Table 18: Parameters for experiment: Number of repetitions of testing packets (1 PREQ per testing packet)

Parameter	Value/Description
	IDS setup
Rules active	Dropping of UDP
Repetitions of testing packets	1, 2, 3, 4, 5, 10
Interval between replications	500 ms
Other information	One PREQ packet is transmitted to the target node before transmitting a testing packet
Packets transmitted per analysis	2 (1 PREQ), 4 (2 PREQs), 6 (3 PREQs), 8 (4 PREQs), 10 (5 PREQs), 20 (10 PREQs)
	Background traffic
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s

Table 19: Parameters for experiment: Number of repetitions of testing packets (2 PREQs per Analysis)

Parameter	Value/Description
	IDS setup
Rules active	Dropping of UDP
Repetitions of testing packets	1, 2, 3, 4, 5, 10
Interval between replications	500 ms
Other information	Two PREQ packets are transmitted to the target node at the beginning of each analysis
Packets transmitted per analysis	3 (2 PREQs), 4 (2 PREQs), 5 (2 PREQs), 6 (2 PREQs), 7 (2 PREQs), 12 (2 PREQs)
	Background traffic
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s

Table 20: Parameters for experiment: Interval between repetitions of testing packets

Parameter	Value/Description
	IDS setup
Rules active	Dropping of UDP
Repetitions of testing packets	5
Interval between replications	100 ms, 300 ms, 500 ms, 1 s, 2 s
Other information	-
	Background traffic
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s

Table 21: Parameters for experiment: Bayes classifier with selective dropping of packets (all tests)

Parameter	Value/Description
	IDS setup
Rules active	Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
Repetitions of testing packets	5
Interval between replications	500 ms
	Attacker node setup
Attacker node	Node B
Attack description	Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
	Background traffic
Traffic throughput	0 (0%) Mbit/s
	Experimentation
Number of analyses (DogoIDS)	10 per variable studied
Replications	3

Table 22: Parameters for experiment: Bayes classifier with selective dropping of packets (only Dropping of SSH packets, and Dropping of UDP packets)

Parameter	Value/Description
	IDS setup
Rules active	Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
Repetitions of testing packets	5
Interval between replications	500 ms
	Attacker node setup
Attacker node	Node B
Attack description	Dropping of SSH, Dropping of UDP
	Background traffic
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s
	Experimentation
Number of analyses (DogoIDS)	30 per variable studied
Replications	3

A.4 EXPERIMENTS OF CHAPTER 7

Table 23 describes the parameters for the detection of the selective dropping of packets. Table 24 shows the configuration for detecting the black hole attack, and Table 25 describes the parameters for the detection of the colluding misrelay attack.

A.4.1 *Setting up the attacks*

Selective dropping of packets

The implementation of the selective dropping of packets is the same as in the previous chapters (refer to Section A.3.)

Black hole attack

To launch the black hole attack, we ran the security testing tool *mdk3* [102]. In collaboration with its developers, we significantly enhanced the tool to support testing of IEEE 802.11s.

Colluding misrelay attack

We implemented the colluding attack in the node A and node B, as it is depicted in Fig. 54. We modified the path table of the node A in order to establish that the next hop address for every destination is the address of node B (MAC address `00:1b:b1:02:00:3f`). Figure 55 depicts the path tables of node A before and after implementing the colluding attack. The path tables were obtained by invoking the command `iw dev <mesh_dev>mpath dump`. We cropped out some of the columns for space reasons. Node B is configured to drop all the packets that come from node A.

DEST ADDR	NEXT HOP	IFACE	SN	METRIC
00:1b:b1:00:ef:9e	00:1b:b1:02:01:12	mesh1	1	16386
00:1b:b1:00:ef:6e	00:1b:b1:00:ef:6e	mesh1	0	8193
00:1b:b1:02:00:3f	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:00:e8	00:1b:b1:02:00:e8	mesh1	0	8193
00:1b:b1:02:01:3f	00:1b:b1:02:01:3f	mesh1	0	8193
00:1b:b1:02:01:0c	00:1b:b1:02:01:0c	mesh1	0	8193
00:1b:b1:02:01:12	00:1b:b1:02:01:12	mesh1	0	8193
00:1b:b1:02:01:23	00:1b:b1:00:ef:6e	mesh1	4	24579
00:1b:b1:02:00:fa	00:1b:b1:02:00:fa	mesh1	0	8193

(a) Original path table.

DEST ADDR	NEXT HOP	IFACE	SN	METRIC
00:1b:b1:00:ef:9e	00:1b:b1:02:00:3f	mesh1	1	16386
00:1b:b1:00:ef:6e	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:00:3f	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:00:e8	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:01:3f	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:01:0c	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:01:12	00:1b:b1:02:00:3f	mesh1	0	8193
00:1b:b1:02:01:23	00:1b:b1:02:00:3f	mesh1	4	24579
00:1b:b1:02:00:fa	00:1b:b1:02:00:3f	mesh1	0	8193

(b) Modified path table for allowing the colluding misrelay attack.

Figure 55: Path table of node A before and after the deployment of the colluding misrelay attack.

Table 23: Parameters for experiment: Detection of the selective dropping of packets attack

Parameter	Value/Description
	IDS setup
Rules active	Black hole, Colluding misrelay, Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
Repetitions of testing packets	5
Interval between replications	500 ms
	Attacker node setup
Attacker node	Node B
Attack description	Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets
	Background traffic
Traffic tool	`iperf`
Traffic type	UDP packets, CBR
Server node	Node C
Server command	`iperf -s -u -i 1`
Client node	Node D
Client command	`iperf -u -c 192.168.0.5 -b 6.25M -t 1000 -i 1` for generating 6.25 Mbit/s
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s
	Experimentation
Number of analyses (DogoIDS)	30 per variable studied
Replications	3

Table 24: Parameters for experiment: Detection of the black hole attack

Parameter	Value/Description
	IDS setup
Rules active	Black hole, Colluding misrelay, Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
Repetitions of testing packets	5
Interval between replications	500 ms
	Attacker node setup
Attacker node	Node B
Attack description	Black hole
	Background traffic
Traffic tool	`iperf`
Traffic type	UDP packets, CBR
Server node	Node C
Server command	`iperf -s -u -i 1`
Client node	Node D
Client command	`iperf -u -c 192.168.0.5 -b 6.25M -t 1000 -i 1` for generating 6.25 Mbit/s
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s
	Experimentation
Number of analyses (DogoIDS)	10 per variable studied
Replications	3

Table 25: Parameters for experiment: Detection of the colluding misrelay attack

Parameter	Value/Description
	IDS setup
Rules active	Black hole, Colluding misrelay, Dropping of ARP, Dropping of HTTP, Dropping of HTTPS, Dropping of only TCP, Dropping of only UDP, Dropping of TCP and UDP, Dropping of DNS, Dropping of SSH, Dropping of all packets, No dropping
Repetitions of testing packets	5
Interval between replications	500 ms
	Attacker node setup
Attacker nodes	Node A and node B
Attack description	Colluding misrelay attack
	Background traffic
Traffic tool	`iperf`
Traffic type	UDP packets, CBR
Server node	Node C
Server command	`iperf -s -u -i 1`
Client node	Node D
Client command	`iperf -u -c 192.168.0.5 -b 6.25M -t 1000 -i 1` for generating 6.25 Mbit/s
Traffic throughput	0 (0%), 6.25 (25%), 12.5 (50%), 18.75 (75%), 25 (100%) Mbit/s
	Experimentation
Number of analyses (DogoIDS)	10 per variable studied
Replications	3

B

CURRICULUM VITÆ

PERSONAL DATA

Name	Rodrigo Daniel do Carmo
Date of birth	20 July 1986
Place of birth	Comodoro Rivadavia, Argentina
Nationalities	Argentine, Portuguese

EDUCATION

since 04/2011	Doctoral candidate at the Department of Computer Science Technische Universität Darmstadt Darmstadt, Germany
03/2005 – 03/2010	Ingeniero en Telecomunicaciones (Telecommunications Engineer) Universidad Blas Pascal Córdoba, Argentina
03/2002 – 12/2004	Técnico en Electrónica (Electronics Technician, High school diploma) Colegio Provincial Técnico 749 Comodoro Rivadavia, Argentina

WORK EXPERIENCE

since 04/2014	Research assistant at the Secure Mobile Networking Lab Technische Universität Darmstadt Darmstadt, Germany
04/2011 – 04/2014	Doctoral scholarship holder at DFG GRK 1362 Technische Universität Darmstadt Darmstadt, Germany
04/2010 – 10/2010	Software security engineer Intel ASDC Córdoba, Argentina

SUPERVISED MASTER AND DIPLOMA THESES

2012	P. Klobuszenski. Mobile phones as sensors for intrusion detection in wireless mesh networks.
2013	T. Hoffmann. Erweiterung eines Intrusion Detection Systems zur Erkennung selektiven Verwerfens von Datenpaketen.
2014	B. Sattler. Implementation and Detection of Colluding Injection Attacks by Means of Active Probing.

Darmstadt, October 14th, 2014

C

AUTHOR'S PUBLICATIONS

CONFERENCES AND WORKSHOPS

- R. do Carmo, J. Hoffmann, V. Willert, and M. Hollick. Making active-probing-based network intrusion detection in wireless multihop networks practical: a bayesian inference approach to probe selection. In *Proceedings of the 39th IEEE Conference on Local Computer Networks (LCN'14)*, 2014.
- R. do Carmo and M. Hollick. Analyzing active probing for practical intrusion detection in wireless multihop networks. In *Proceedings of the 11th IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (WONS'14)*, 2014.
- R. do Carmo and M. Hollick. DogoIDS: a mobile and active intrusion detection system for IEEE 802.11s wireless mesh networks. In *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec'13)*, 2013.
- R. do Carmo, M. Werner, and M. Hollick. Signs of a bad neighborhood: a lightweight metric for anomaly detection in mobile ad hoc networks. In *Proceedings of the 8th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet'12)*, 2012.
- R. do Carmo, M. Nassar, and O. Festor. Artemisa: an open-source honeypot back-end to support security in VoIP domains. In *Proceedings of the 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, 2011.

POSTERS, EXTENDED ABSTRACTS AND TECHNICAL REPORTS

- R. do Carmo, M. Werner, P. Klobuszenski, and M. Hollick. Not the enemy but the ally: increasing the security of wireless networks using smartphones. In *Proceedings of the 7. Essener Workshop Neue Herausforderungen in der Netzsicherheit*, 2013.
- R. do Carmo and M. Hollick. DogoIDS: a mobile and active intrusion detection system for IEEE 802.11s wireless mesh network. *Technical Report TR-SEEMOO-2012-04*, Technische Universität Darmstadt, October 2012.
- R. do Carmo, M. Nassar, and O. Festor. A honeypot back-end to support security in VoIP domains (Poster and Demo). In *IEEE Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 2010.